SPEECH RECOGNITION USING THE MELLIN TRANSFORM

THESIS

Jesse R. Hornback, Second Lieutenant, USAF

AFIT/GE/ENG/06-22

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GE/ENG/06-22

SPEECH RECOGNITION USING THE MELLIN TRANSFORM

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Electrical Engineering

Jesse R. Hornback, B.S.E.E.

Second Lieutenant, USAF

March 2006

AFIT/GE/ENG/06-22

SPEECH RECOGNITION USING THE MELLIN TRANSFORM

Jesse R. Hornback, B.S.E.E.
Second Lieutenant, USAF

Approved:

/signed/	
Dr. Steven C. Gustafson (Chairman)	date
/signed/	
Dr. Richard K. Martin (Member)	date
/signed/	
Dr. Timothy R. Anderson (Member)	date
/signed/	
Dr. Raymond E. Slyh (Member)	date

AFIT/GE/ENG/06-22

Abstract

The purpose of this research was to improve performance in speech recognition. Specifically, a new approach was investigating by applying an integral transform known as the Mellin transform (MT) on the output of an auditory model to improve the recognition rate of phonemes through the scale-invariance property of the Mellin transform. Scale-invariance means that as a time-domain signal is subjected to dilations, the distribution of the signal in the MT domain remains unaffected. An auditory model was used to transform speech waveforms into images representing how the brain "sees" a sound. The MT was applied and features were extracted. The features were used in a speech recognizer based on Hidden Markov Models. The results from speech recognition experiments showed an increase in recognition rates for some phonemes compared to traditional methods.

Acknowledgments

I would like to express my sincere appreciation to my faculty advisor, Dr. Steven Gustafson, for his guidance and support. The many hours he spent helping me accomplish this project were crucial in the success of this thesis effort. I would also like to thank my sponsors, Dr. Tim Anderson and Dr. Ray Slyh, from the Air Force Research Laboratory (AFRL/HECP) for both the support and time spent helping me in this endeavor. I am also indebted to Dr. Richard Martin for his help and input amidst a busy schedule.

I would also like to thank my lovely wife for her constant support and encouragement throughout my time at AFIT and for gracefully enduring all the hours I spent studying and working.

Last, but not least, I would like to thank my Lord and Savior Jesus Christ. I can do all things through Him who gives me strength.

Jesse R. Hornback

Table of Contents

List of Figures

List of Tables

Speech Recognition Using the Mellin Transform

I. Introduction

Speech recognition has many military and commercial applications, for example: hands-free voice control of cockpit or automotive controls; voice-based data entry for applications that would normally require several mouse clicks or have several text entry fields; telephony applications such as telephone banking, catalogue centers, and call routing for customer service centers; and voice-based biometrics, etc. Because of the wide applicability of speech recognition, it has received a great deal of research for a number of decades [1]. Despite this considerable attention, automatic speech recognizers still do not perform as well as human for most tasks.

One problem in speech recognition is achieving good speaker-independent performance. A speech recognizer trained on many examples of speech for a given individual can often perform well. However, a recognizer trained on the same amount of data but from a wide range of speakers, usually does not perform nearly as well. There are a number of reasons why this is the case. For example, women and children tend to have shorter vocal tracts than men, leading to shifts in the formants (vocal tract resonances). Also, women and children tend to have higher average pitch than men. Another reason is the different accents and dialects among speakers. These various differences among speakers cause considerable variability in the standard features used in speech recognition, which in turn reduces the phoneme discrimination of a recognizer, where phonemes are basic elements of speech. This research attempts to partially address this speaker-independent speech recognition problem through the use of a feature set based on an auditory model and the Mellin transform.

The Mellin transform (MT) [2] is the integral transform

$$M_{f(t)}(s) = \int_0^\infty t^{s-1} f(t)\, dt, \quad s \in \mathbb{C}. \tag{1}$$

The usefulness of the MT lies in its scaling property. Research has shown that the MT normalizes vowel feature sets from speakers with different voice pitches [3]. The benefits of the MT with an auditory model are that it helps to separate pitch information from the vocal tract configuration and that it generates a representation that separates the vocal tract size information from the general shape information. The research conducted here uses an entirely new approach in the field of speech recognition by performing the MT on all speech data, not just vowels, to determine if features from the MT lead to improved phoneme discrimination across speakers.

For this research several Matlab scripts were written to run experiments and to supplement previously written code. Altering part of the code that executes the MT resulted in a reduction in computation time by a factor of four. Hidden Markov models (HMMs) were used to perform recognition experiments. The results were compared to results from traditional automatic speech recognition (ASR) using the standard features, which are mel frequency cepstral coefficients (MFCCs). The results obtained from the speech recognition experiments show a recognition rate improvement for some phonemes over conventional methods used in ASR.

Chapter 2 discusses the terms and the basic tools used in this research and provides a background for understanding the methodology. Chapter 3 discusses the experimental methodology, including steps for obtaining results and why each step was taken. Chapter 4 analyzes the results, and Chapter 5 provides a discussion and recommendations for future research.

II. Background

This chapter discusses the basics of speech recognition and also defines the terms and tools used to accomplish the results of this research. Once a background understanding of the methods for speech recognition is reached, the experimental methodology discussed in Chapter 3 will be understood more completely.

2.1 Speech Recognition Basics

As mentioned in the previous chapter, speech recognition is highly challenging in that it requires developing statistical models to understand and recognize human speech. The basic model for a speech recognition system is shown in Figure 1. The first step is to extract features from the speech that will be used in the pattern recognition analysis to recognize the speech. Successful speech recognition requires prior knowledge in the form of an acoustic model, a pronunciation dictionary, and a language model. The recognizer uses the prior knowledge sources and the feature vectors to determine a set of words according to the fundamental speech recognition equation [4] given as:

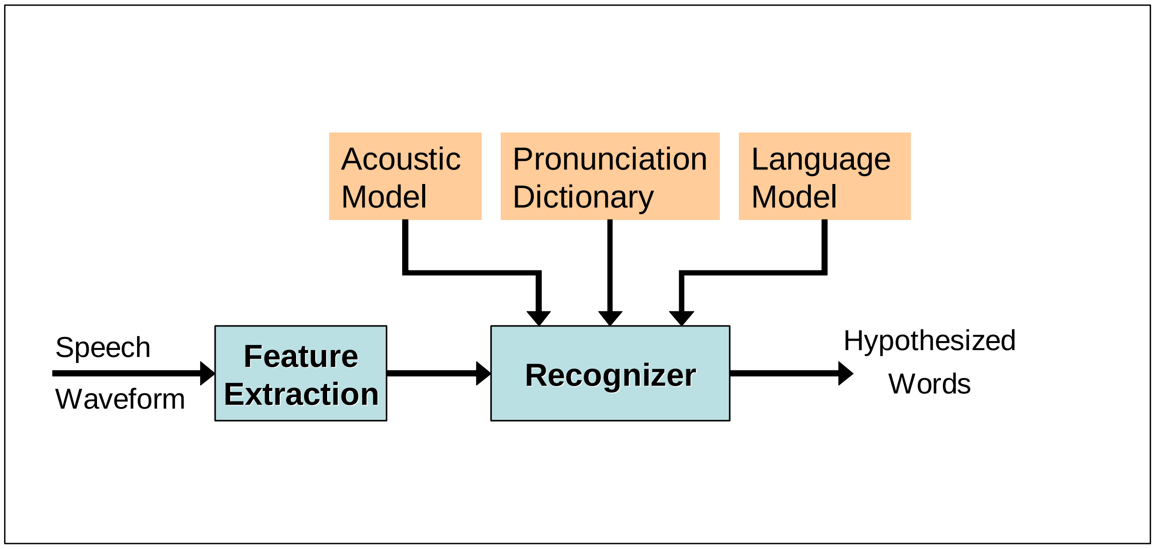

Figure 1. A block diagram of the basic model for speech recognition.

$$\underline{w}' = \arg\max_{\underline{w}} P\left(\underline{w} \middle| X\right). \tag{2}$$

This equation states that the hypothesized words, w', equal the argument that maximizes the probability of the words, w, given the acoustical features matrix X. Using Bayes' rule, this equation becomes

$$\underline{w}' = \arg\max_{\underline{w}} \frac{P\left(X \middle| \underline{w}\right) P\left(\underline{w}\right)}{P\left(X\right)}. \tag{3}$$

The probability P(X) of the feature matrix is simply a scalar constant for all word sequences, so it can be ignored. This leaves the following equation to describe the speech recognition process:

$$\underline{w}' = \arg\max_{\underline{w}} P\left(X \middle| \underline{w}\right) P\left(\underline{w}\right) \tag{4}$$

The P(w) term is the prior probability of a sequence of words, w, which is described by the language model. Language models are one component that a speech recognizer uses and are discussed in further detail below in Section 2.2. The P(X|w) factor is the probability of a feature matrix given the word sequence, w. This term is taken into account through the acoustic models.

The final component for a speech recognizer is a pronunciation dictionary. An example of part of a pronunciation dictionary is shown in Table 1. The pronunciation dictionary tells the recognizer how words are broken up into smaller units called phonemes, which are the smallest basic units of speech. There are about 39 different phonemes in the English language. Speech recognition is often performed using phoneme-level acoustic models rather than word-level models. This is due to a lack of data necessary for training individual word models. This research uses phoneme-level acoustic models.

ABBREVIATE	[ABBREVIATE]	AH B R IY V IY EY T SP
ABBREVIATE	[ABBREVIATE]	AX B R IY V IY EY T SP
ABDOMEN	[ABDOMEN]	AE B D OW M AH N SP
ABDOMEN	[ABDOMEN]	AE B D AX M AX N SP
ABIDES	[ABIDES]	AH B AY D Z SP
ABIDES	[ABIDES]	AX B AY D Z SP
ABILITY	[ABILITY]	AH B IH L AH T IY SP
ABILITY	[ABILITY]	AX B IH L IX T IY SP
ABLE	[ABLE]	EY B AH L SP
ABLE	[ABLE]	EY B EL SP

Table 1. Part of a typical pronunciation dictionary used for speech recognition. The first column is the word to be recognized, the second column is the output when that word is recognized, and the third column shows a breakdown of all the phonemes that make up each word, where "SP" denotes a short pause.

2.2 Language Models

Language models estimate the probability of sequences of words [4]. A speech recognizer uses a language model to estimate the probability a given word will follow another word in a spoken sequence. Common language models are bigram and trigram models. These models contain computed probabilities of groupings of two or three particular words in a sequence, respectively. This project uses a phoneme-level language model for phoneme recognition experiments. The phoneme-level language model used in this research, allows any phoneme to follow any other phoneme with equal probability. Language models are not the focus of this research and therefore are not discussed further.

2.3 Hidden Markov Models

HMMs are the acoustic models that produce the best results in speech recognition [5]. They estimate probabilities of sequences of events, and are comprised of states, where each state determines a set of probabilities. In general, HMMs are described by a state transition matrix, where each state has a transition probability of moving from the current state to the next state and also has probability densities of emitting continuous features from each state. An example of a three state HMM transition matrix is shown in Figure 2, where each number is a probability of going from the current state, represented by the rows, to the next state in the sequence, represented by the columns. For example, the value in row 2, column 3 represents the probability of moving from state 1 to state 2. The first row represents the initial or starting state, and the last row represents the final or exit state, which do not emit anything. The initial state is just a starting point, so the HMM cannot remain in the initial state and cannot return to the initial state once it is left, so the probabilities are zero for all of column 1. Similarly, once the exit state is reached, the HMM cannot transition to any other state, so the probabilities in row 5 are all zero.

In Figure 3 the arrows represent the probabilities governed by the state transition matrix. The *i* and *e* states are the initial and exit non-emitting states, and states 1, 2, and 3 are the emitting states. HMMs may use as many states as desired, although one to five is the norm for speech recognition. The more states that are used, the more complex an HMM model becomes, due to the fact that more parameters must be calculated to describe it. The parameters in the state transition matrix and the state emission probability densities begin with initial guesses, and training provides more accurate estimates of these parameters. The algorithm that accomplishes training by iteratively

$$\begin{array}{cc|ccccc|}
 & & \multicolumn{5}{c}{\text{next state}} \\
 & & i & 1 & 2 & 3 & e \\
 & i & 0.0 & 1.0 & 0.0 & 0.0 & 0.0 \\
\text{current state} & 1 & 0.0 & 0.6 & 0.4 & 0.0 & 0.0 \\
 & 2 & 0.0 & 0.0 & 0.6 & 0.4 & 0.0 \\
 & 3 & 0.0 & 0.0 & 0.0 & 0.7 & 0.3 \\
 & e & 0.0 & 0.0 & 0.0 & 0.0 & 0.0
\end{array}$$

Figure 2. An example of a state transition matrix of an HMM.

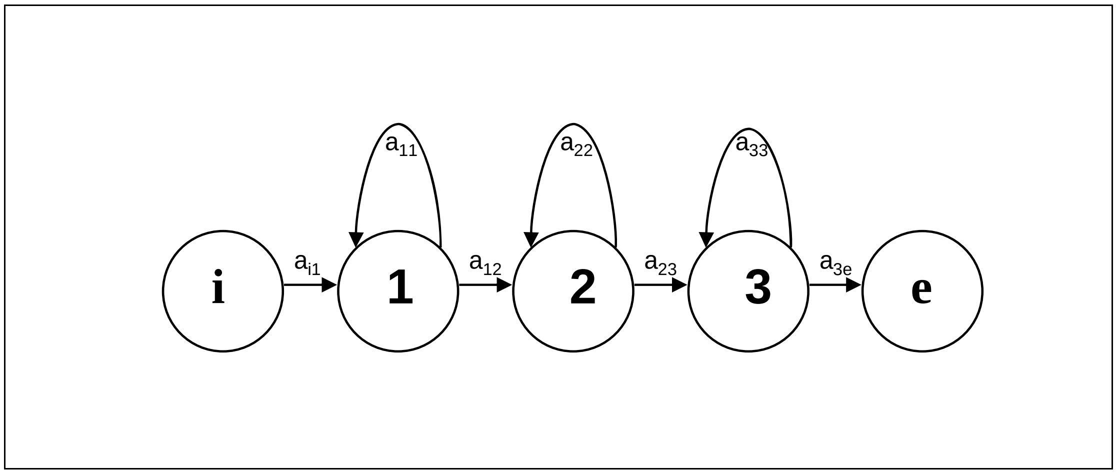

Figure 3. Diagram of a 3 state HMM with initial state and end state. The *a* variables represent state transition probabilities.

estimating and re-estimating these parameters is known as the Baum-Welch algorithm [4]. The Baum-Welch algorithm, also known as the Forward-Backward algorithm [5], is an expectation maximization algorithm that works iteratively to update the parameters of the HMMs to match the observed sequence of training data.

This research uses continuous density HMMs, which use a Gaussian probability density for each state to model the probability distribution of emitting continuous observation vectors from the HMM. The means and variances of these Gaussian mixture densities are estimated by the Baum-Welch algorithm for continuous HMMs using Equations 5-11. The probability of generating observation $\mathbf{v}_t$ in state j [4] is computed using Equation 5.

$$b_j\left(\mathbf{v}_t\right) = \sum_{m=1}^{M_j} c_{jm} N\left(\mathbf{v}_t; \mu_{jm}, \Sigma_{jm}\right), \tag{5}$$

where M_j is the number of mixture components in state j, c_{jm} is the weight of the m'th component and $N(\mathbf{v}_t; \mu, \Sigma)$ is a multivariate Gaussian density with mean vector μ and covariance matrix Σ, *i.e.*,

$$N\left(\mathbf{v};\mu,\Sigma\right)=\frac{1}{\sqrt{\left(2\pi\right)^{n}\left|\Sigma\right|}}e^{-\frac{1}{2}\left(\mathbf{v}-\mu\right)'\Sigma^{-1}\left(\mathbf{v}-\mu\right)}, \tag{6}$$

where n is the dimensionality of **v** [4] and |Σ| denotes the determinant of the matrix Σ.

Next, the forward probability of observing the speech vectors while in state j at time t is estimated using Equation 7

$$\alpha_j\left(t\right)=\left[\sum_{i=2}^{N-1}\alpha_i\left(t-1\right)a_{ij}\right]b_j\left(\mathbf{v}_t\right), \tag{7}$$

where the state transition probability is a_{ij} [4]. The first and last states are the initial and exit states which do not emit; hence the limits of the summation do not include those states. Next, the backward probability [4] is estimated using Equation 8.

$$\beta_i\left(t\right)=\sum_{j=2}^{N-1}a_{ij}b_j\left(\mathbf{v}_{t+1}\right)\beta_j\left(t+1\right), \tag{8}$$

where $\beta_i(t)$ is the probability that the model is in any state and will generate the remainder of the target sequence from time = t + 1 to time = T [5]. The transition probabilities [4] are then able to be estimated using Equation 9.

$$a'_{ij}=\frac{\sum_{t=1}^{T}\alpha_i\left(t\right)a_{ij}b_j\left(\mathbf{v}_{t+1}\right)\beta_j\left(t+1\right)}{\sum_{t=1}^{T}\alpha_i\left(t\right)\beta_i\left(t\right)} \tag{9}$$

Equations 10-11 describe the calculations for estimating the means and variances of the Gaussian mixtures. Each observation is weighted by $L_j(t)$, which is the probability of being in state j at time t, and normalized by dividing by the sum of all the L_j probabilities [4].

$$\mu'_j=\frac{\sum_{t=1}^{T}L_j\left(t\right)\mathbf{v}_t}{\sum_{t=1}^{T}L_j\left(t\right)} \tag{10}$$

$$\Sigma'_j = \frac{\sum_{t=1}^{T} L_j(t)\left(\mathbf{v}_t - \mu_j\right)\left(\mathbf{v}_t - \mu_j\right)'}{\sum_{t=1}^{T} Lj(t)} \tag{11}$$

The set of equations described above can be used iteratively as many times as needed to get better estimates of the HMM parameters. In general, there is an equation for estimating the c_{jm} terms; however, this research used only single-mixture models for each state so the c_{jm} terms were all unity. For the speech recognition experiments performed here, the HMMs are trained with features extracted from the speech phonemes. Thousands of feature vectors are used as training data, and the result is one HMM that represents each individual phoneme.

Once the HMMs are trained, testing may begin. Testing classifies unknown phonemes by finding which HMM phoneme model is most likely to have produced the observed features. The test speech data are decoded using the HMMs along with a language model and a dictionary. Various algorithms exist for decoding the test speech data. The one employed in this project is known as the Viterbi algorithm [4]. The Viterbi algorithm is a dynamic programming algorithm that calculates the most likely set of HMM states that produced the observed set of sequences, which in this case is the test data, taking into account a language model and dictionary for computing results. For example, the algorithm takes a test speech input and computes the sequence of HMM phoneme models most likely to have produced it [5].

2.4 Mel Frequency Cepstral Coefficients as Features

Features often used for training HMMs are MFCCs [1] [4]. These are coefficients based on the Mel scale that represent sound. The word cepstral comes from the word cepstrum which is a logarithmic scale of the spectrum (and reverses the first four letters in the word spectrum). Figure 4 illustrates how MFCCs are calculated. First, the speech

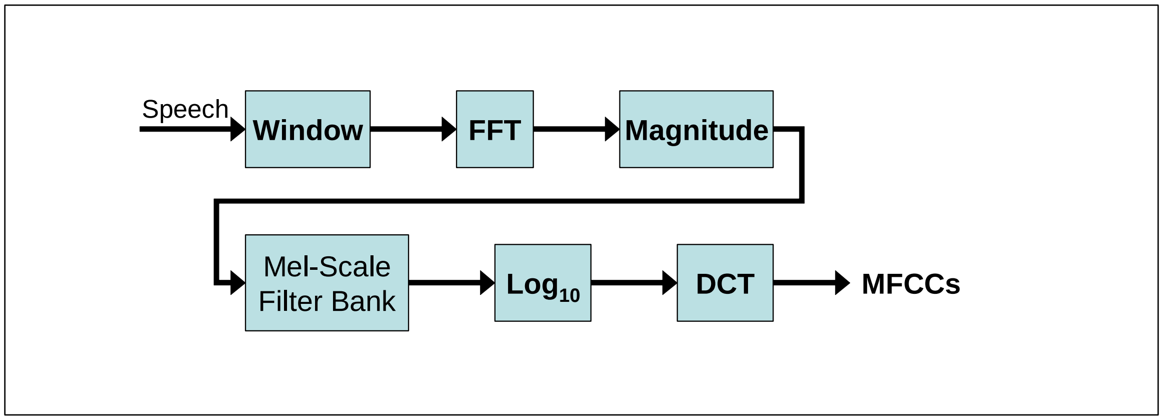

Figure 4. Block diagram of the calculations for MFCCs.

data are divided into 25 ms windows (frames). A new frame is started every 10 ms making this the sampling period and causing the windows to overlap each other. Next, the fast Fourier transform is performed on each frame of speech data and the magnitude is found. The next step involves filtering the signal with a frequency warped set of log filter banks called Mel-scale filter banks. These log filter banks collect the signal information into the coefficients m_i, which are the log filter bank amplitudes. The log filter banks are arranged along the frequency axis according to the Mel scale, a logarithmic scale that is a measure of perceived pitch or frequency of a tone [6], thus simulating the human hearing scale. The Mel scale is defined in Equation 12.

$$Mel(f) = 2595 \log_{10}\left(1 + {}^{f}\!/_{700}\right) \tag{12}$$

The Mel scale yields a compression of the upper frequencies where the human ear is less sensitive. The filtering process is illustrated in Figure 5. Next, the logarithm is taken of the log filter bank amplitudes. Finally, the MFCCs are calculated using the discrete cosine transform (DCT) in Equation 13.

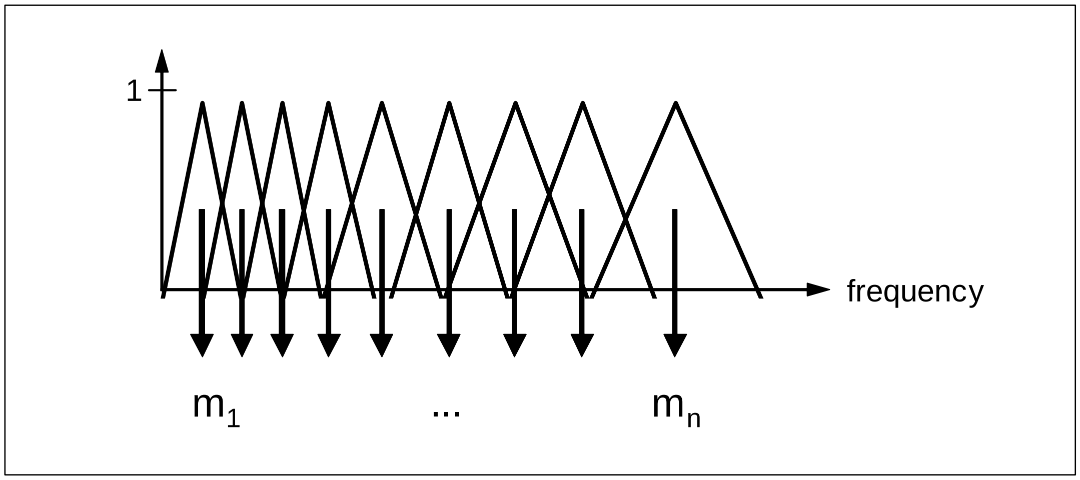

Figure 5. An example of a Mel-scale filter bank.

$$c_i = \sqrt{\frac{2}{N}} \sum_{j=1}^{N} m_j \cos\left(\frac{\pi i}{N}\left(j - 0.5\right)\right), \tag{13}$$

where N is the number of filter banks and the c_i terms are the resulting MFCCs.

To further enhance speech recognition performance, an extra set of delta and acceleration coefficient features are sometimes calculated with MFCCs. These features are the first and second time derivatives of the original coefficients, respectively.

The results obtained in this project are compared to speech recognition performance on regular MFCCs as well as on MFCCs with delta and acceleration coefficients. Generally, the MFCC method for ASR yields the most successful results.

2.5 Auditory Image Model

An alternative method of feature extraction that is a more recent development than MFCCs is something called the Auditory Image Model (AIM). The AIM model is software that models how the human ear processes speech [7]. It models the human hearing mechanism by simulating the processes the ear performs on a sound, resulting in an auditory image that represents the sound. AIM includes tools to simulate the spectral analysis, neural encoding, and temporal integration performed by the auditory system [7].

Figure 6 shows a block diagram that represents the process that AIM uses to process a sound. The PCP (pre-cochlear processing) block performs filtering of the signal to represent the response up to the oval window of the inner ear. The BMM (basilar membrane motion) block represents the basilar membrane motion response to the signal. It is simulated by a gamma-tone filter bank of bandpass filters with evenly distributed center frequencies along a quasi-logarithmic scale known as an equivalent rectangular bandwidth (ERB) scale [8]. This process transforms the signal (in effect) to a moving surface that represents the basilar membrane as a function of time [9]. The NAP (neural activity pattern) block simulates the neural activity pattern produced by basilar membrane energy transduction to the auditory nerve which generates its firing activity pattern [7].

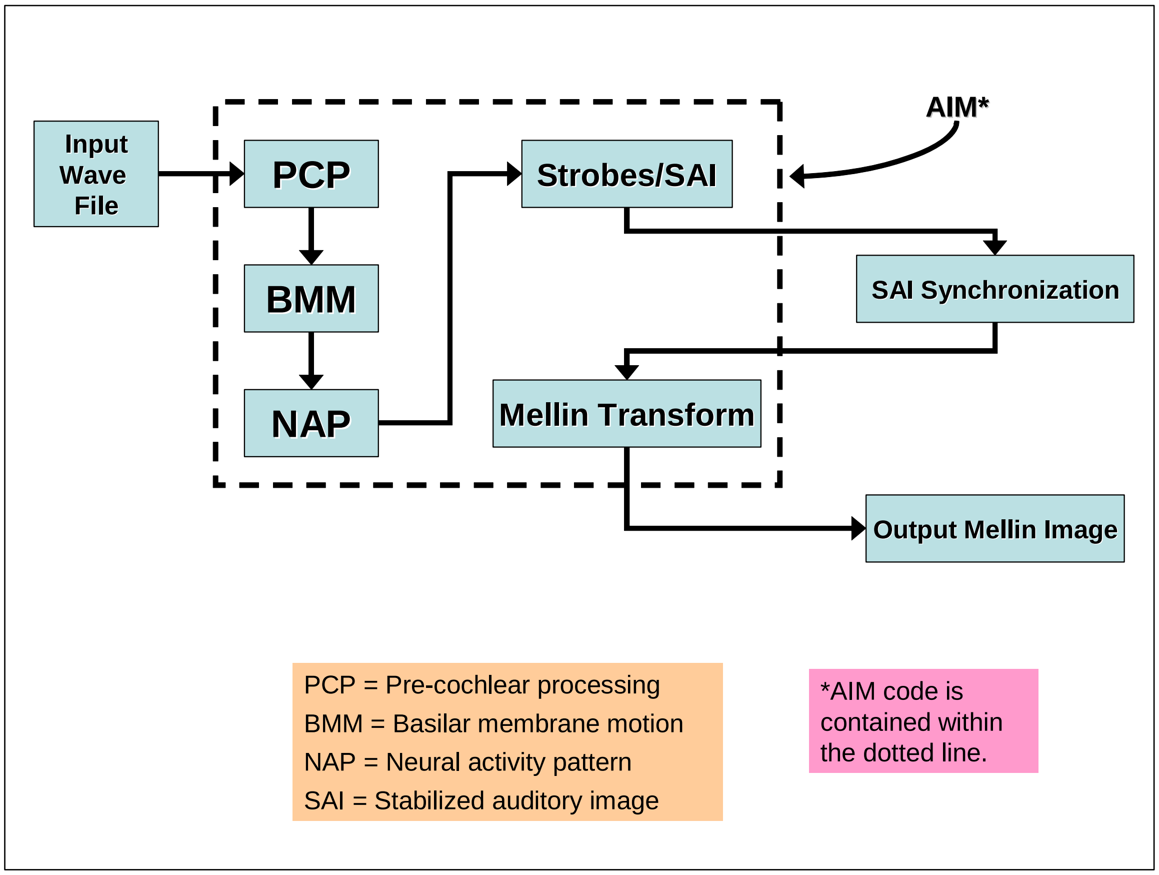
Input Wave File
PCP
BMM
NAP
Strobes/SAI
Mellin Transform
AIM*
SAI Synchronization
Output Mellin Image
PCP = Pre-cochlear processing
BMM = Basilar membrane motion
NAP = Neural activity pattern
SAI = Stabilized auditory image
*AIM code is contained within the dotted line.

Figure 6. A block diagram of AIM and each of its modules.

Figure 7 shows one frame of the NAP of a speech signal. The pulses have a rightward skew from high to low frequency (as shown by the dotted lines) because of the phase lag in the output of the cochlea in the BMM [7].

The Strobes/SAI (stabilized auditory image) block calculates the stabilized auditory image using strobed temporal integration (STI) and represents the temporal integration performed on the NAP. This process simulates the perception of the human ear by stabilizing oscillating sounds into static patterns [7]. STI works by strobing the

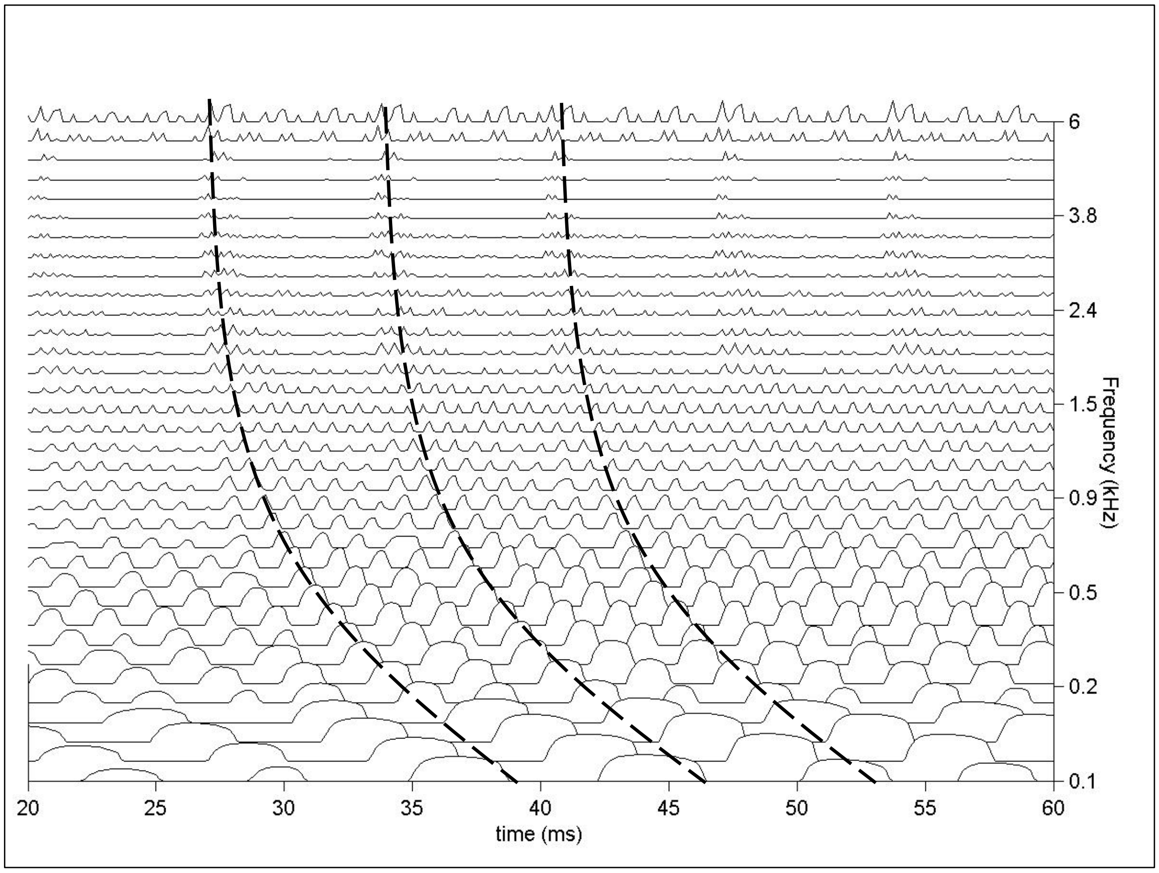

Figure 7. A frame of the NAP of a vowel sound generated by the NAP block of Figure 6 and presented as a waterfall plot. The NAP module of AIM converts basilar membrane motion into a representation that is expected to be similar to the pattern of neural activity found in the auditory nerve or cochlea nucleus. The abscissa of the plot is time, and on the ordinate each horizontal line represents one of 35 channels with center frequencies from 100 Hz to 6 kHz on a log scale. The height of each pulse represents the "firing rate" of the NAP.

signal to the levels of high activity in the NAP by locating the points in each channel that are local maxima. This information then defines the limits of integration when performing temporal integration for calculating the SAI [8]. Figure 8 shows one frame of the SAI of the speech signal from Figure 7. The SAI representation is based on the assumption that as the NAP flows from the cochlea, the human hearing mechanism acts as a bank of delay line filters [10] that capture information into a buffer store. This process stabilizes the repeating patterns of the NAP into the SAI, which is an image representing the sound. The phase lag from the NAP plot is removed, causing the phase

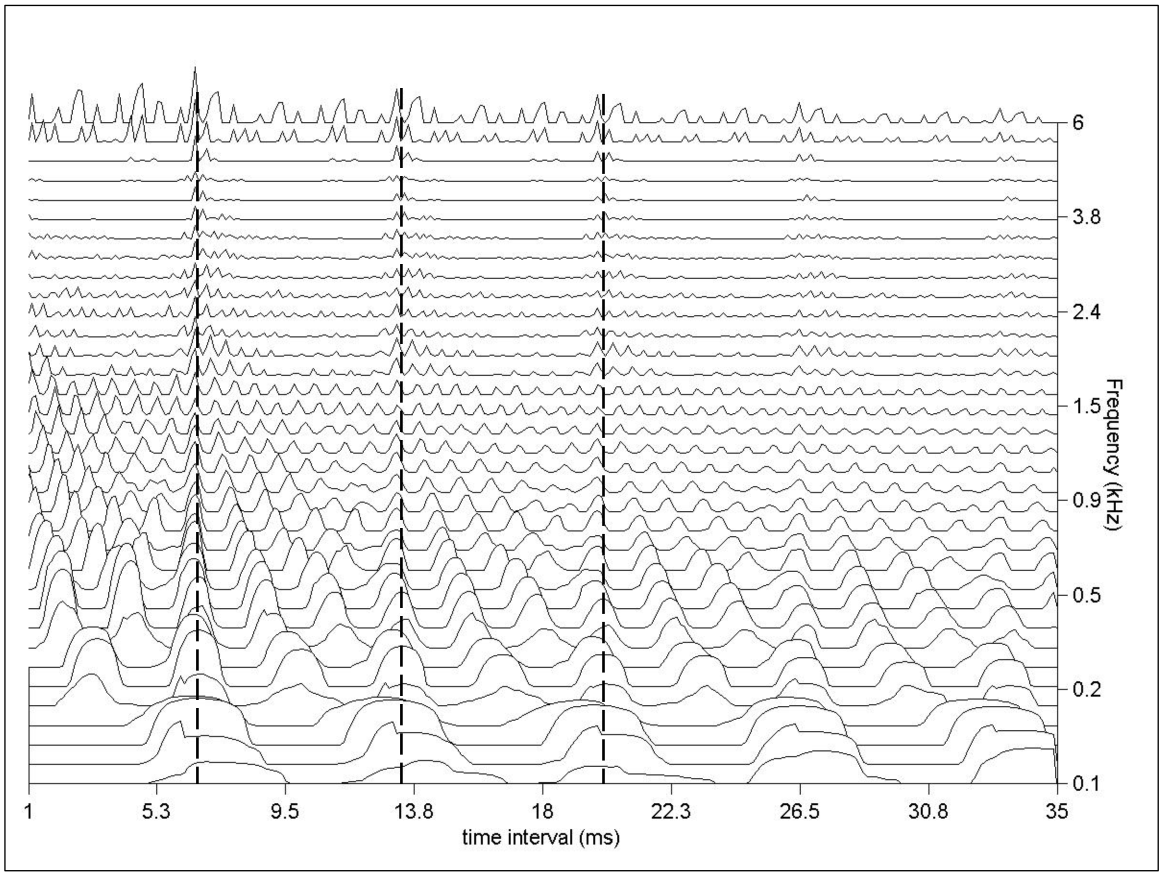

Figure 8. A frame of the stabilized auditory image (SAI) of a vowel sound generated by the AIM. The SAI module of AIM uses STI to convert the NAP into the SAI. The abscissa of the plot is time, the ordinate is frequency on a log scale, and vertical height represents the "firing rate" of the SAI.

to be aligned in the SAI plot as shown by the dotted lines. The SAI is described by Equation 14 as

$$A_I\left(\alpha f_0,\tau\right)=\sum_{k=0}^{\infty}S_w\left(\alpha f_0,\tau+kt_p\right)e^{-\xi\tau}e^{-\eta kt_p}\ , \tag{14}$$

where S_w is the output of the NAP, αf_0 is the peak frequency of each auditory filter in the filter bank, τ is the time axis for the SAI, t_p is the period of the signal, and η and ξ are factors that affect the time interval of each SAI frame and the decay rate of the waveforms in each frame [11]. The pattern of the pulse peaks or ridges in the SAI follow a time-interval-peak frequency product path, denoted by *h*, which is constant along the ridges of the SAI. This time interval-peak frequency product path is used later in the calculation of the MT. The SAI synchronization block and its justification are discussed in Section 3.2.2.

2.6 The Mellin Transform and its Applications

The MT is the integral transform defined in Equation 1. Similar to the Fourier transform (FT), the MT possesses certain properties [12], some of which are displayed in Table 2. As mentioned previously, the scaling property of the MT is exploited in this

Property	**Function**	**Mellin Transform**
Standard	$f(t)$	$M(s)$
Scaling	$f(at)$	$a^{-s}M(s)$
Linear	$af(t)$	$a\,M(s)$
Translation	$x^a f(t)$	$M(a+s)$
Exponentiation	$f(t^a)$	$a^{-1}\,M(s/a)$

Table 2. Key properties of the MT. The property of interest here is the scaling property, which states that dilation of the abscissa in the time-domain by the factor *a* has no effect on the shape of *M(s)* in the Mellin-domain. The time dilation by *a* is encoded in the a^{-s} factor and does not dilate *M(s)*.

project. The FT is translation-invariant in that it does not matter if the signal is shifted in time by some *Δt*; the magnitude of the FT of the signal remains the same (although the phase changes). This translation invariance does not hold for the MT, however, so the limits of integration must be defined when it is calculated. The limits of integration are defined by the STI, as discussed in the previous section. The MT is not translation invariant, but it has a scaling property, and when evaluated under certain conditions it is scale invariant within a phase factor [11] [3]. This scale invariance comes from the scaling property of the MT and means that as the time-domain distribution of the signal is subjected to dilation, the magnitude distribution of the MT does not dilate. Contrast this to the FT, where if the time axis of a signal is compressed or expanded, the magnitude of the Fourier spectrum is expanded or compressed, respectively. However, for the MT, dilation of the time-axis does not compress or expand the distribution in the Mellin domain. Equations 15-17 show how the scaling property of the MT, affects time axis dilation of a signal. In particular, let

$$M_{f(t)}(s) = \int_0^\infty t^{s-1} f\left(t\right) dt \,. \tag{15}$$

If the time axis is dilated by a factor of *a*, the result is

$$M_{f(at)}(s) = \int_0^\infty t^{s-1} f\left(at\right) dt \,. \tag{16}$$

With $\tau = at$, or $t = \tau/a$, the integral is:

$$\int_0^\infty \left(\frac{\tau}{a}\right)^{s-1} f\left(\tau\right) \frac{d\tau}{a} \quad = \quad \left(\frac{1}{a}\right)^{s-1} \left(\frac{1}{a}\right) \int_0^\infty \tau^{s-1} f\left(\tau\right) d\tau \quad = \quad a^{-s} M_{f(t)}\left(s\right). \tag{17}$$

Thus, if *M(s)* is the MT of some function *f(t)*, the introduction of a dilation factor *a* either expands or compresses the time axis, but in the Mellin domain the net result of the MT is a segregation of the size and shape information from the signal [11]. This normalizing effect of the MT may be useful for achieving improved speaker independent speech recognition. Figure 9 shows the MT of the SAI from Figure 8. Chapter 3 gives an in-depth description of how the MT is calculated in Matlab, which indicated that the MT is evaluated using an FT such that *s = -jc + ½*.

The MT has many uses, including digital image registration and digital watermarking [13], digital audio effects processing [14], and vowel normalization [11]. Experiments in vowel normalization have shown positive results [3]. Thus, this research attempts to use the normalizing effect of the MT on all speech to determine if results improve over conventional ASR methods.

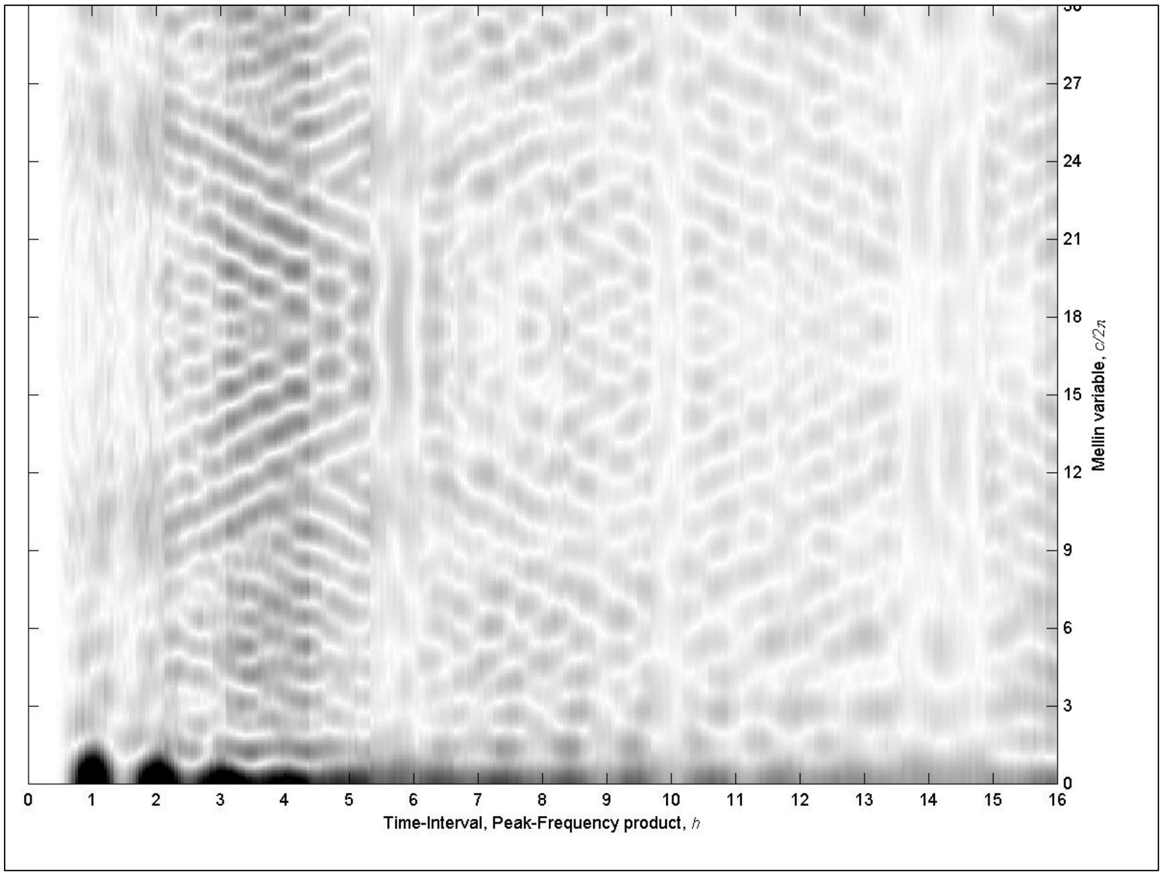

Figure 9. A section of the MT of the SAI from Figure 8 generated by the MT block of the AIM. The abscissa of the plot is called the time-interval-peak frequency product because the ridges in the SAI plot follow a path where the time-interval and the peak frequency in the SAI equal a constant, *h*. The MT is computed along these vertical paths, which results in a two-dimensional MT plot. The ordinate shows the Mellin variable, which is the argument of the MT, much like *jω* in the FT. Darker color in a region indicates greater Mellin coefficient magnitude.

III. Experimental Design

This section presents the steps taken for phoneme recognition experiments with MFCCs and the MT features using HMMs trained with the Hidden Markov Model toolkit (HTK) [15]. The experiments were run using the TIMIT database.

3.1 TIMIT Database

The TIMIT database is a collection of 6300 sentences, 10 sentences each spoken by 630 persons from eight different dialect regions in the United States. The speech was recorded at Texas Instruments (TI) and transcribed at the Massachusetts Institute of Technology (MIT), which is how the database derives its name. It was created for the purpose of "providing speech data for the acquisition of acoustic-phonetic knowledge and for the development and evaluation of automatic speech recognition systems" [16]. The database is divided into training data, which consists of approximately 73% of the database, and testing data, which consists of approximately 27% of the database. The transcription files containing the words and phonemes of each spoken sentence are also contained in the database.

For some experiments, a subset of the database is used instead of the entire database. This subset consists of 100 sentences, 10 spoken utterances each from 10 speakers. For the subset, training is performed by the leave one out method which trains on 9 speakers and tests on the 10th speaker, then leaves a different speaker out for testing, etc. The results of leaving each speaker out once is averaged and shown as the result. The reason for using this subset was to obtain some results quickly from each of the experiments before performing them on the entire database, which requires much more time.

TIMIT is a small sized database by current standards, but it is a good choice for this research because it is a diverse collection of data that possesses a good balance between having enough data for training and testing and being small enough to avoid an exorbitant amount of time in running experiments. It also is useful for speech recognition research because it has been in existence for over two decades, has been used in numerous experiments, and can be used for comparison with results obtained here.

3.2 Mellin Transform Processing

Several versions of the AIM exist. For this research, the Auditory Image Model in Matlab (*aim-mat*) [7] is used to implement the AIM, because it is the only version that includes code for performing the MT. *Aim-mat* has a graphical user interface (GUI) which allows a user to load a sound file and perform AIM calculations on it with ease; however, this research employs the command line version of *aim-mat* instead of the GUI to receive input and produce results so that batch processing can be implemented without user interaction. A parameter structure file called "all_parameters.m" (listed in Appendix A) includes all necessary parameters for the AIM calculations. When run in Matlab, this structure file creates a variable called "all_options", which specifies various parameters and options necessary for each block of AIM.

3.2.1 Processing up to the SAI Stage

Figure 10 shows the overall process. The first step is to convert the speech files in the TIMIT database, which are NIST sphere files, to wave file format so that Matlab can use them. This conversion is accomplished using the Perl script "convert_timit_wav.pl" (listed in Appendix B). The Matlab script "main_program.m" (listed in Appendix C) was developed to use the *aim-mat* code by calling the appropriate functions, processing the resulting data from these functions, and writing results to the

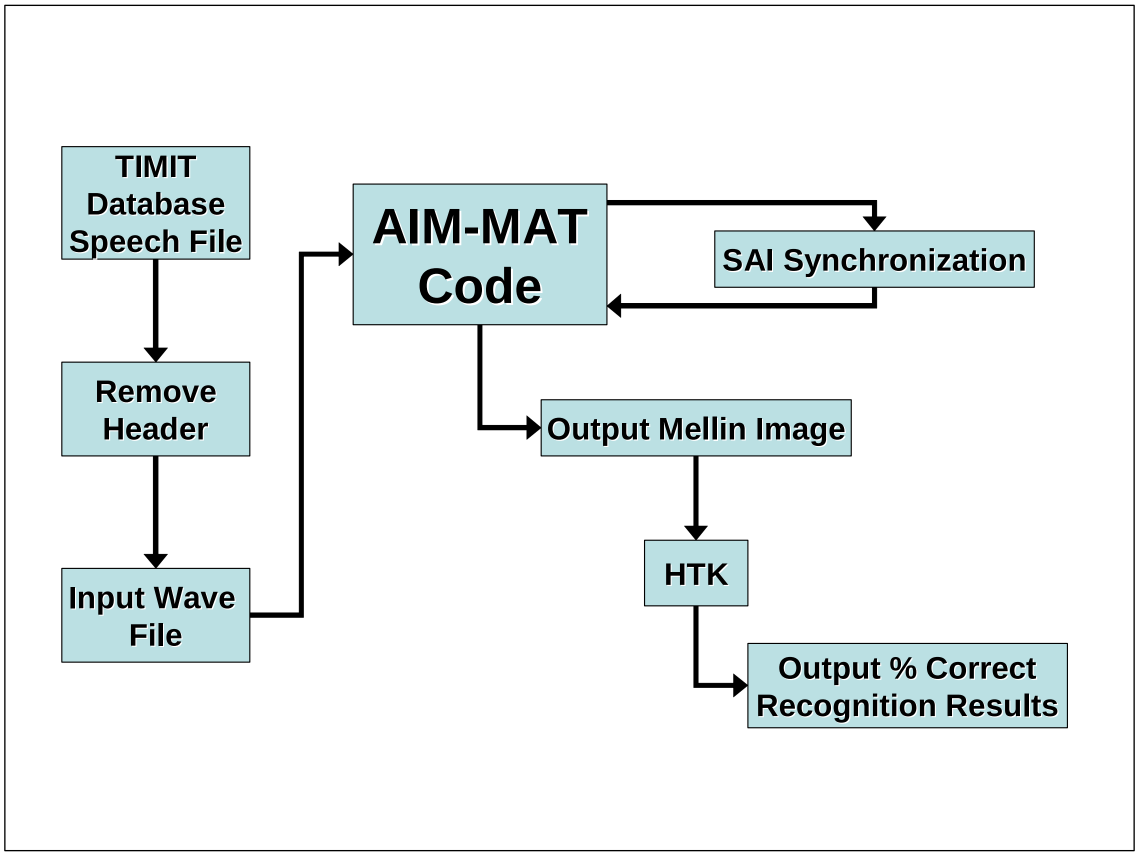

Figure 10. Block diagram of the overall procedure. Data from the TIMIT database is first preprocessed so that it is useable in Matlab. The aim-mat code uses the resulting input wave files to construct the SAI and the MT of the SAI. A modification made to the code written specifically for this research is represented by the SAI synchronization block. This module converts the SAI, which is asynchronous, into a synchronous data stream before the MT is computed and the MT images are sent to HTK for recognition tests. The HMMs require synchronous data for training.

correct location. Thus, the "main_program.m" script runs the entire process of the conversion from wave files to Mellin transform data files, including loading the parameter file, calling the *aim-mat* subroutines, loading SAI and MT image data, saving SAI and MT image data, plotting, and converting MT image data to HTK format. The paths in the "main_program.m" script must be set correctly to read from the directory where the wave files are stored and to write the SAI and MT results in the desired directory. The options in the "main_program.m" script can be set so that only the SAI is calculated, only the MT is calculated (on previously calculated SAI), both SAI and MT are calculated on the fly, or the HTK conversion of the previously calculated MT files is calculated. The parameters for each of the six AIM modules are specified in the "all_parameters.m" script. Some of the parameters include specifying the algorithms used in the AIM modules. Each module can use different algorithms to accomplish the calculation. The built-in algorithms used here are:

- PCP: none
- BMM: gamma-tone filter bank
- NAP: irinonap
- Strobes: irinostrobes
- SAI: irinosai
- User-Module: mellin

These algorithms perform the steps in the AIM (as discussed in Section 2.5), as well as the MT. They were chosen because they are less computationally intensive than alternative algorithms. Future research could investigate the effects of different choices for the various components.

Once all the parameters and options are set, the "main_program.m" script uses the *aim-mat* code to generate the AIM of the speech signals, which includes its SAI and the MT of the SAI. The SAI is a representation of the speech signal that is divided into frames, where each frame represents 35 ms of the speech signal.

3.2.2 SAI Synchronization

The SAI representation is asynchronous, meaning that the time intervals between the start-times of each frame are not the same. Later, HMMs are used to perform the speech recognition experiments, and they require synchronous input data. Therefore, the

SAI must be synchronized prior to the experiments. The SAI synchronization block from Figure 6 represents this step, which is performed before the MT calculation. The script called “synch_sai.m” (listed in Appendix D) is additional code, written specifically for this research, to execute the SAI synchronization process. It works by sampling the SAI every 10ms and taking the frame that starts the closest to each 10 ms sampling period. The sampling rate of 10 ms was chosen because it equals the sampling rate used for calculating MFCCs. By using the same sampling rate for both MFCC calculation and SAI synchronization calculation, results can be more accurately compared between both methods. This SAI synchronization process discards some of the original frames, which is acceptable because portions of each of the frames representing the speech signal overlap. Each frame contains an image with six to eight vertical pulse patterns called strobes, where each strobe contains a decayed version of the previous strobe. The synchronization processes each frame by taking only the first 10 ms, which is approximately the first two strobes of each SAI frame, because each frame contains a decayed part of the frame previous to it. By removing the decayed portions from each frame, the synchronized SAI contains less redundant information than the original SAI. This action also has the benefit of a reduction in computation time for the MT calculation. Figure 11 shows a synchronized version of the SAI.

3.2.3 Mellin Transform Calculation

After the necessary synchronization process is complete, the final step in the *aim-mat* code performs the calculations for the MT, and the results are saved to the previously specified path in the “main_program.m” script.

Aim-mat uses the MT to map auditory speech images from vocal tracts of different sizes into an invariant MT image [17] by performing a one dimensional MT

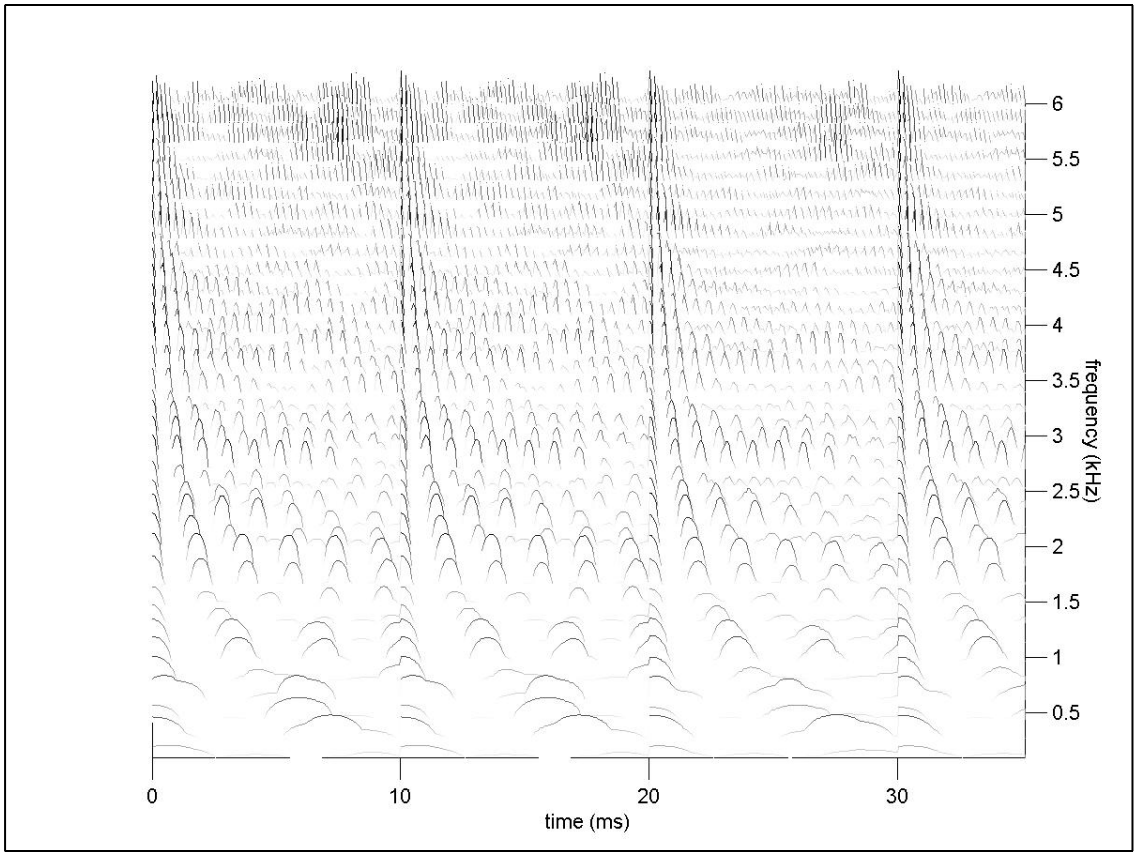

Figure 11. The sampled version of the SAI generated by the "SAI synchronization" code. Each strobe begins at each multiple of 10 ms, which is due to the 10 ms sampling rate of the synchronization process. The process also takes the first two strobes from each frame and removes the rest, which is a decayed version of the first 2 strobes. The abscissa of the plot is time, the ordinate is frequency on a log scale, and vertical height represents the "firing rate" of the SAI.

along each time interval-peak frequency product column of the SAI, resulting in a collection of MTs which form a two-dimensional image. The MT of the SAI [11] from Equation 14 is

$$M\left(s,h\right)=\int_0^T A_I\left(\frac{h}{\tau},\tau\right)e^{(s-1)\ln\tau}d\tau\,, \tag{18}$$

where A_I is the SAI representation and h is the parameter representing the time interval-peak frequency product constant.

Calculation of the MT image from the SAI is accomplished by a two stage process. First, a time dilation of each of the SAI channels by a factor proportional to the center frequency of the channel filter is performed. This intermediate representation is called the size-shape image (SSI) [17] and implements the ln τ term in Equation 18 by creating a log-time axis [11]. The logarithmic time scale is achieved if we let

$$t = e^x, \quad dt = e^x dx, \quad 0 \rightarrow -\infty, \quad \infty \rightarrow \infty .$$

This causes the form of the MT when evaluated at $s = -jc + ½$ to be

$$\left. \begin{aligned} M_{f(t)}\left(-jc+1/2\right) &= \int_0^\infty e^{-jc-x-1/2} f\left(e^x\right) e^x \, dx \\ &= \int_0^\infty e^{-jcx} e^{-1/2} f\left(e^x\right) dx \\ &= \frac{1}{e^{1/2}} \int_0^\infty e^{-jcx} f\left(e^x\right) dx \end{aligned} \right\}, \tag{19}$$

which is the FT on a logarithmic time scale. This is used in the second state where the SSI is used to compute the MT. The center frequencies of the AIM filters are now on a logarithmic scale along the abscissa. This coordinate system makes the MT equivalent to a FT on spatial frequency [11], where each column of the final MT image is computed by performing the FT on the columns of the SSI [17] and taking the magnitude.

3.3 Illustration of the Effect of Pitch Scaling

Generally women and children have higher pitched voices than men. Figures 12 and 13 use this fact to illustrate the effects of pitch scaling on the MT by showing a simulated vowel sound at pitches of 100 Hz and 200 Hz, respectively. This pitch difference simulates how typical male and female voices produce the same vowel sound with different pitches. The first, second, and third formants, indicated by the horizontal arrows in both figures, are at a frequency of 450 Hz, 1450 Hz, and 2450 Hz, respectively, simulating the formants of the phoneme /uh/. It can be seen that the pitch periods for the male spoken vowel of Figure 12 last approximately twice the amount of time as those for the female spoken vowel of Figure 13, as indicated by the vertical arrows in each figure.

Figures 14 and 15 show the MT of Figures 12 and 13, respectively. It can be seen that the high amplitude regions for Figure 14 lie in the same regions as those of Figure 15. The differences in the amplitudes are due to the fact that the two vowels are not fully scaled versions of each other. Only the pitch is scaled; the formants are the same for the two signals.

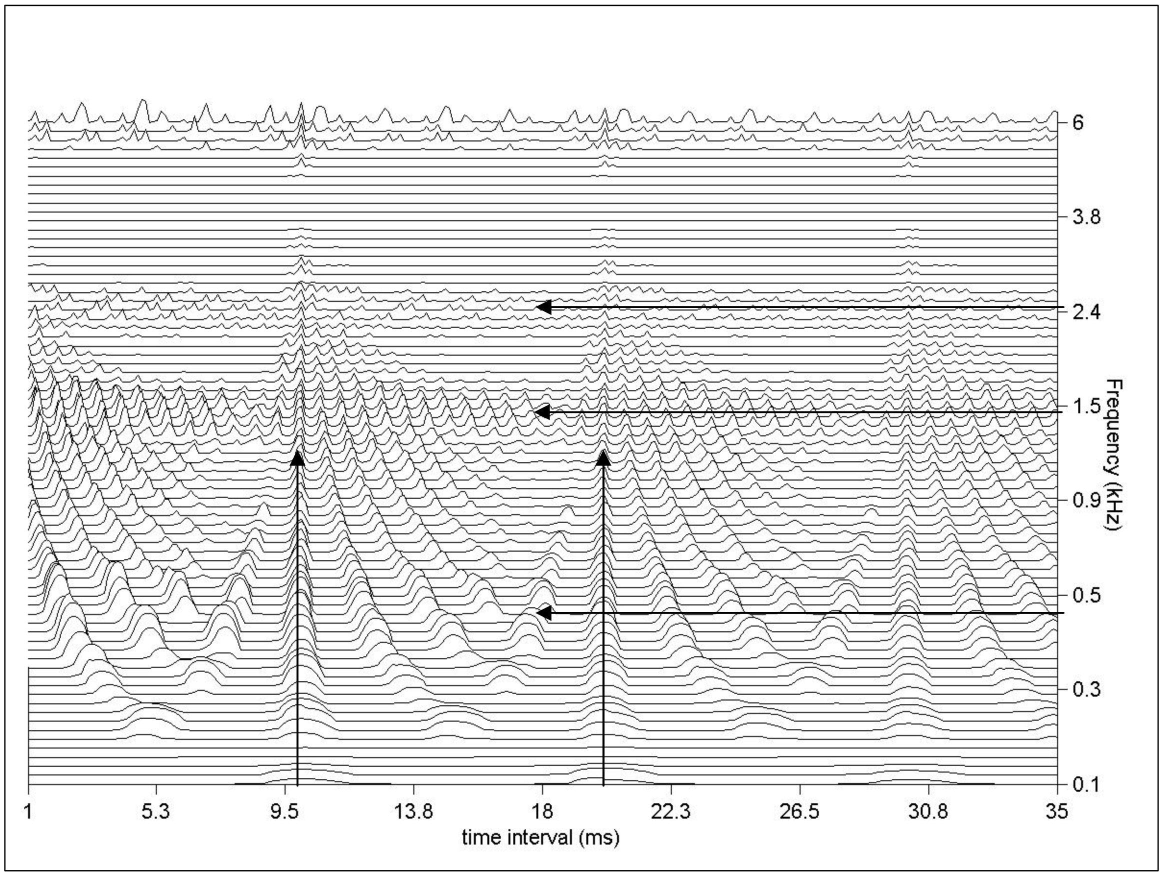

Figure 12. The SAI of the vowel sound /uh/ at a pitch of 100 Hz, typical of a male speaker. Notice that the pitch period is 10 ms as indicated by the vertical arrows. The vowel formants are 450 Hz, 1450 Hz, and 2450 Hz as shown by the horizontal arrows. Compare this figure to Figure 13, which shows the same vowel spoken with a pitch of 200 Hz, typical of a female speaker.

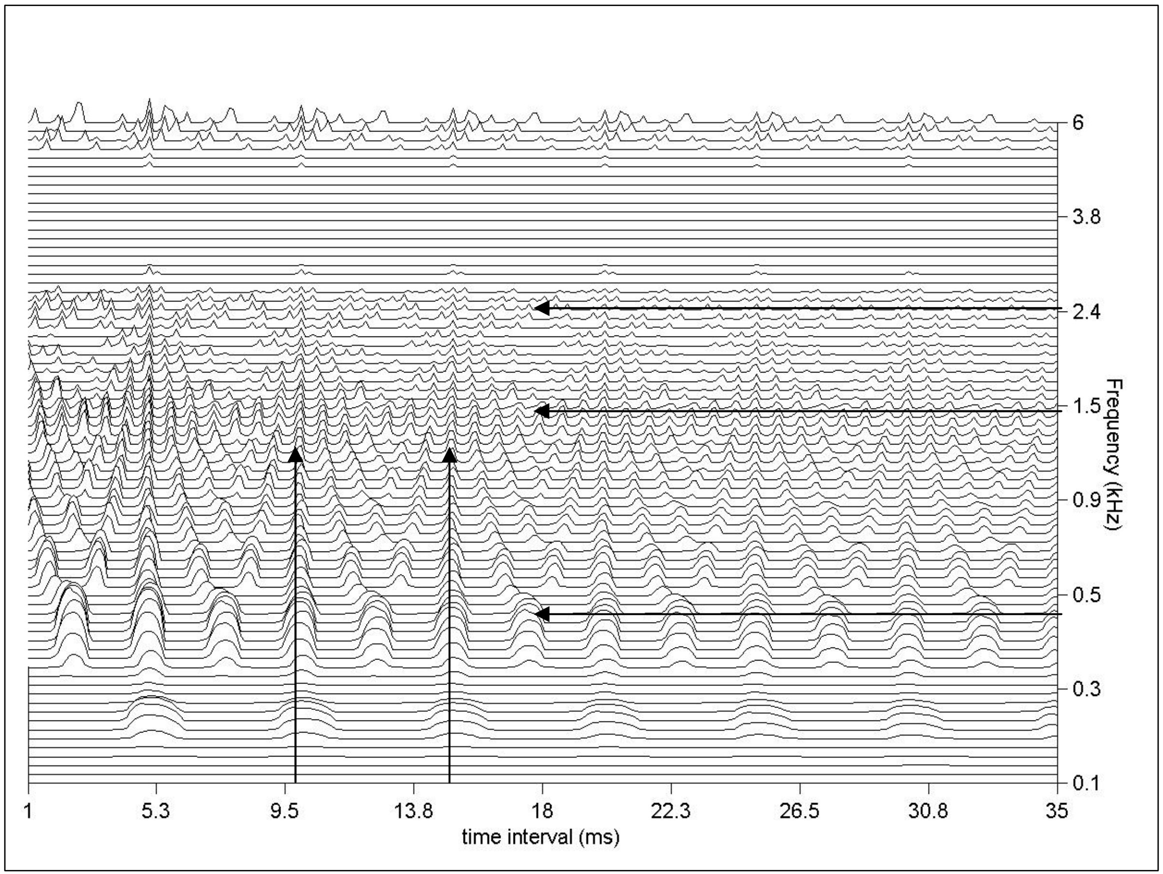

Figure 13. The SAI of the vowel sound /uh/ at a pitch of 200 Hz, typical of a female speaker. Notice the pitch period is 5 ms as indicated by the vertical arrows.

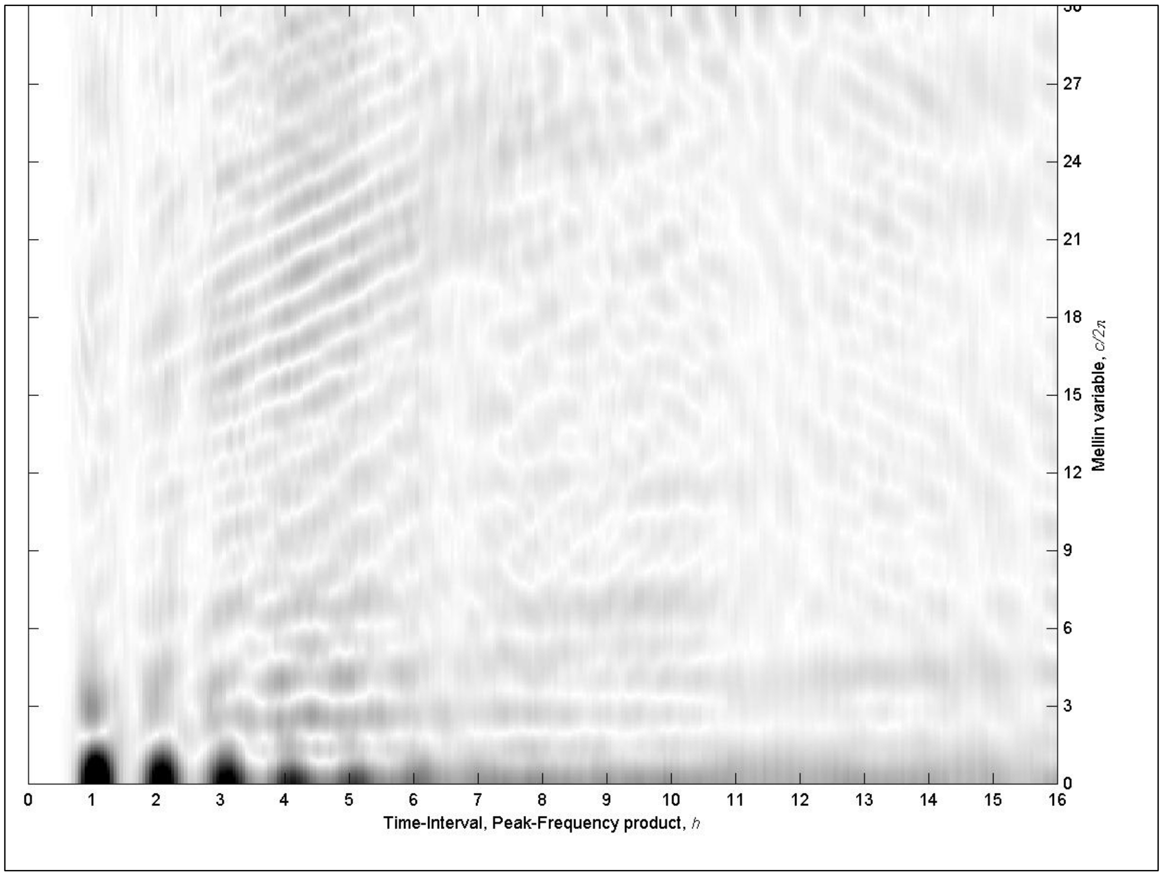

Figure 14. The MT of the SAI from Figure 12. Compare this plot to Figure 15, which is the MT of the SAI from Figure 13.

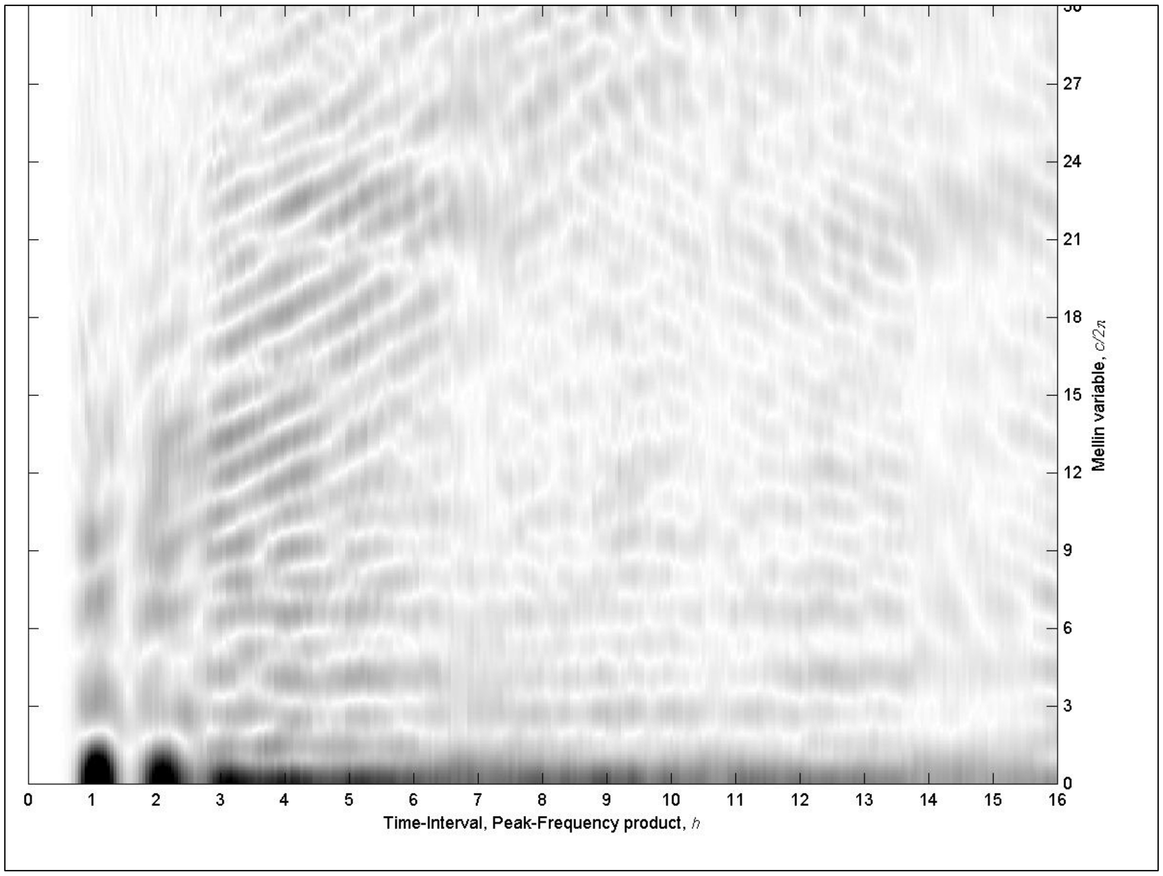

Figure 15. The MT of the SAI from Figure 13. Figures 12 and 13 show considerable changes between male and female vowels, whereas Figures 14 and 15 show similar regions of high amplitude.

3.4 HMMs with HTK

This research uses a set of programs called HTK to train HMMs and conduct phoneme recognition experiments. HTK is open source software designed by the Cambridge University Engineering Department along with Entropic Research Laboratories [15]. HTK also provides utilities for extracting features from speech data and custom user defined features can be imported into HTK, as was the case for this project.

Before the MT data can be used in HTK to perform speech recognition experiments, it must be converted to a format that HTK can use. The Matlab script “writehtk.m”, obtained from the Imperial College department of electrical and electronic engineering at the University of London website [18], is used to perform the conversion of MT data to HTK format. This conversion is accomplished by reshaping the matrices of each MT frame into a row vector so that each frame is one row. The data is written as a floating point binary file. The frame period, which is 10 ms, is also encoded into the file along with a flag identifying the data as user defined features. After this process, the data can be imported into HTK.

Figure 16 illustrates the processes and commands used to perform phoneme recognition. The HCopy command is part of the data preparation phase and calculates MFCC features of the input speech data. Because data preparation for the MT is accomplished prior to importing into HTK, the HCopy command is not needed for the MT features and is omitted. The next command in HTK is HCompV, which works by computing the global mean and covariance of the training data and assigning these values as the starting points in the Gaussians in each phoneme HMM model. This assignment is known as the initialization stage for “flat-start” training, because each phoneme model

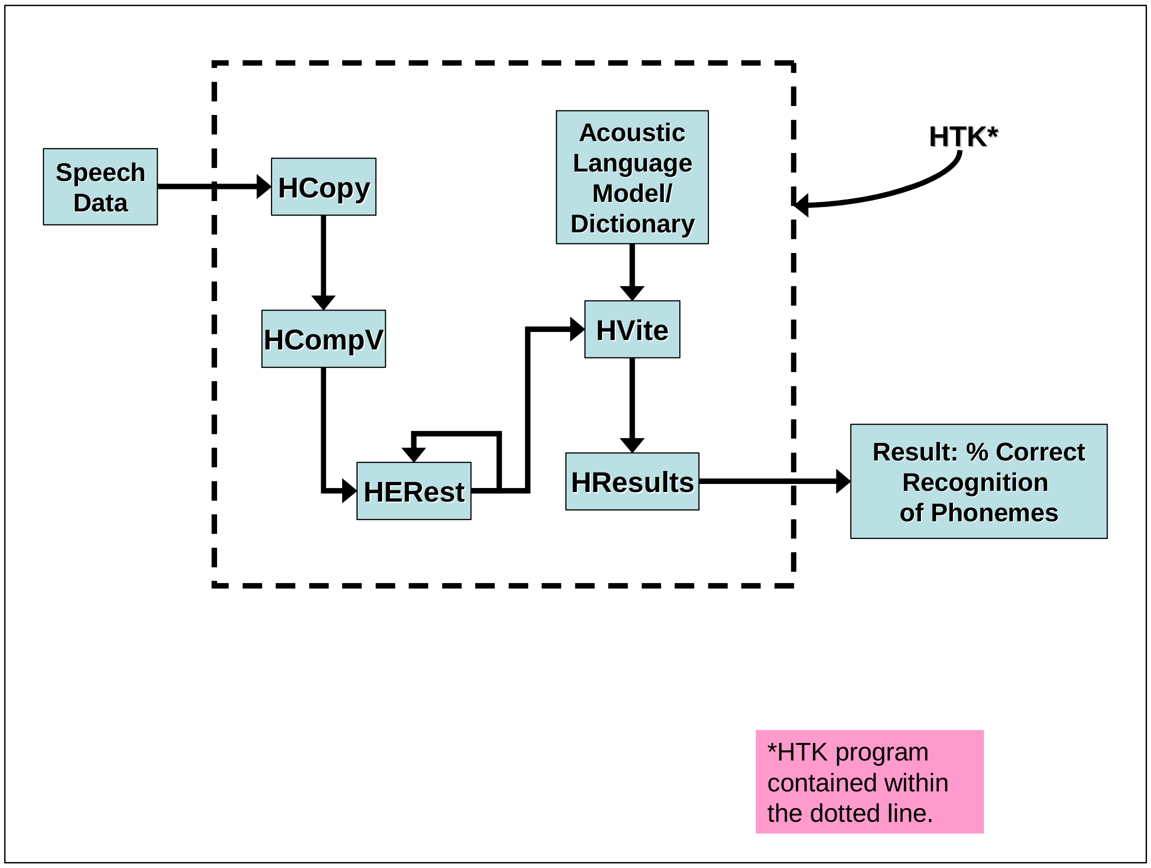
Speech Data
HCopy
HCompV
HERest
Acoustic Language Model/ Dictionary
HVite
HResults
HTK*
Result: % Correct Recognition of Phonemes
*HTK program contained within the dotted line.

Figure 16. A block diagram for steps HTK uses in taking an input and computing percent correct recognition results. Results can be shown in percent correct recognition per sentence, per speaker, or over the entire test data set.

starts identically [4]. Once initial estimates for the HMM models are calculated, the models can be re-estimated by training with the HERest command. This command uses the Baum-Welch algorithm to perform re-estimation of the mean, variance and state transition parameters of the HMMs [4] and may be used iteratively to re-estimate the parameters of the HMM models. For this research, the re-estimation is performed for a total of three iterations. In iteratively estimating the HMM parameters, there is the option of updating the weights, means, and variances of the mixtures. For some of the experiments, a global variance was used and not further updated.

When training is complete, the HVite command is used for testing. HVite is a Viterbi word recognizer that computes the likelihood of the phonemes that produced the given speech data using the HMM models, a language model, and a dictionary, and it outputs a transcription of the speech file. When testing is complete, the HResults command outputs the results, including the percent correct recognition rates as well as other statistics determined by options in the command. Some of these statistics include: the number of correctly recognized phonemes, words, and sentences as well as the number of errors from deletions, insertions, and substitutions. Also, HResults can compute a confusion matrix, which is a matrix that displays all phonemes in rows that show which phoneme is recognized and the number of times it occurs.

Several variations of test data are used for the training and test process in this project. By default the MT produced by the AIM has too many features to estimate with HTK (about 192,000); however the options in the "main_program.m" file can be set to control the MT image data resolution. The number of features and the size of the MT image file are equivalent to image resolution, where a larger resolution means that more pixels are required to describe the image, resulting in a larger memory requirement. Smaller images reduce the memory size of the image but also reduce the information contained in the image, as illustrated in Figure 17, which show the two Mellin image resolutions used here compared to the default resolution that aim-mat produces. HTK was recompiled to accept a maximum number of features of 8192. Due to this limitation, the original number of features set in the options file for the MT data was 8100. The use of 500 features from the MT was also investigated, to make the process of training and testing faster and to observe the impact of the reduced resolution on the recognition results. The

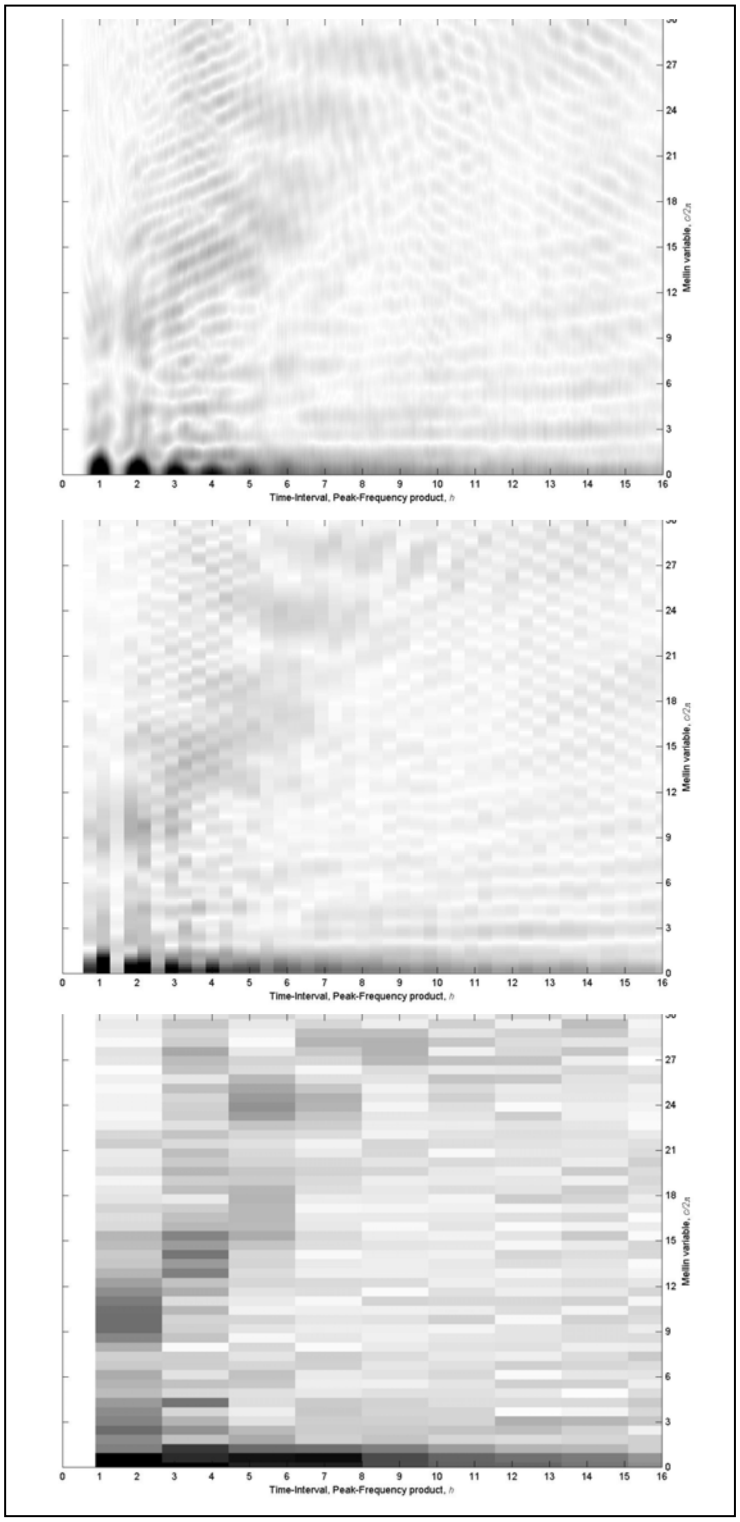

Figure 17. A comparison of the two Mellin image resolutions used in this research compared to the default resolution that aim-mat produces. The top plot is the default *aim-mat* resolution which has about 192,000 features, the middle plot is the one containing 8100 features, and the bottom plot is the one containing 500 features.

SAI data was also used to train and test HMM models with HTK; the SAI data used 5600 features.

IV. Results

Results from each experiment in HTK were output to a master label file (MLF) for analysis by different methods. One method outputs the percent correct recognition of phonemes over the entire test set. Percent correct recognition is defined as H/N x 100%, where H is the number of correctly identified phonemes and N is the total number of phonemes. The number of correctly identified phonemes is given by: $H = N - S - D$, where D is the number of deletions and S is the number of substitutions. A deletion occurs when a phoneme should have been recognized but was omitted. A substitution occurs when a phoneme is mistaken for a different phoneme. Another output statistic is the percent accuracy defined as (H - I)/N x 100%, where I is the number of insertions in the output. An insertion occurs when a phoneme is mistakenly inserted into a place where no phoneme should be recognized. Note that the percent accuracy can be negative if enough insertions occur. Most of the results emphasized below are percent correct recognition of phonemes.

4.1 Overall Results

Table 3 shows overall percent correct recognition results from the experiments described in Chapter 3. In this table, the MFCC, MT, and SAI methods and also the

Method	Global Var	# HMM States	# Feat	# Param	Database Sub-set (100 sentences)	Entire Database (6300 sentences)
MFCC	N	3	13	106	40.23%	41.56%
MFCC_D_A	N	3	39	262	50.94%	54.46%
Mellin	N	3	8100	48628	18.85%	14.11%
Mellin	Y	3	8100	48628	20.22%	18.41%
Mellin	N	1	8100	16212	26.23%	23.68%
Mellin	Y	1	8100	16212	30.62%	27.19%
Mellin	N	3	500	3028	16.67%	10.66%
SAI	N	3	4480	26908	N/A	15.55%

Table 3. Comparison of the results of the HTK experiments shown in percent correct recognition. Note that the number of HMM states is the number of emitting states.

varying parameters used in the experiments are compared for percent correct recognition. Since no delta and acceleration coefficients are calculated for the MT data, it is appropriate to compare them to ordinary MFCC results instead of MFCC with delta and acceleration coefficients. This method is included in the table for reference and to observe how the calculation of delta and acceleration coefficients improves percent recognition results. The method column of the table indicates the method of feature extraction: MFCC, MFCC including delta and acceleration coefficients, the MT images at two different resolutions, and finally the SAI image representation. The global variance column indicates whether the initial global variance computed is the same one used throughout all the HMM re-estimations or if the variance for each HMM model is re-estimated during each iteration. The number of HMM states column indicates the number of states used to build the HMM models. As discussed in the previous section, the number of features for the MT data is restricted by HTK to a maximum of 8192; therefore, the maximum number of features used in the MT experiments is 8100. The number of features column indicates how many features each method uses. The number of parameters column indicates the number of parameters that the HMM models must be trained to estimate. The formula for calculating the number of parameters is

$$\#\,parameters = 2\cdot\left(\#\,states\right)\cdot\left(\#\,features\right)+\left(\#\,states\right)+\left(\#\,states+2\right)^{2} \qquad (20)$$

Diagonal covariance matrices were used, so each state must have means and variances equal to the number of features, which is the reason for the factor of two. Adding the number of states is necessary because a global constant is estimated for each state. The final term is a result of the estimation of the state transition matrix. The state transition matrix includes the initial and final state, hence the +2 in this term. The database subset (100 sentences) column shows results in percent correct recognition when a subset of the

TIMIT database is used for training and testing. The final column shows the results in percent correct recognition when the entire TIMIT database is used for training and testing.

The most interesting aspect of the results in Table 3 is the percent correct recognition in each category. The best performing method for MT data recognition is the 1-state HMM with global variances used throughout HMM re-estimations, with a correct recognition rate of 27.2%. It is obvious that the recognition rates are better when the global variances are used in re-estimations than when the variances are re-estimated in each iteration. The reason for this is that the database is not large enough to support accurate training of the individual mixture variances. Also, note that the 1-state models have roughly one-third the number of parameters to estimate compared to the 3-state models. This reduction in the number of parameters for the HMM models to train makes training with the given data faster and produces better performance, but it also yields less flexibility to the models.

Another interesting result is the decrease in performance when the MT data with 500 features is used. The reduction in features results in a reduction in the MT image resolution, which causes a decrease in percent recognition performance. Even though the number of parameters to train is reduced for the MT image with 500 features, the negative effects from a reduced resolution of the MT image outweigh the positive effects of having fewer parameters to train. Since HTK does not accept more than 8192 features, possible performance improvement using more than 8100 features was not explored, but this possibility would be an interesting topic for future research. Also, a lot more data would be required to estimate significantly more parameters.

The SAI data, used in computing the MT data, was also tested and found to perform better than the corresponding case with the MT. This result suggests that some of the parameter settings for the MT and the SAI synchronization component used in this research should be further investigated.

One final observation to take note of from this table is the fact that performance decreases going from the tests on the database subset to tests on the entire database. This result might be due to the properties of the reduced set of speakers, but more research is needed to determine its cause.

A comparison of the MFCC results with the MT and SAI results does show that the MFCCs have better performance, but they have received considerable research and the research on the MT and SAI is just beginning.

4.2 Individual Phoneme Results

To compare performance for individual phonemes, tables 4-6 show the confusion matrices for results from MFCCs (3-state HMMs), MT with 1-state HMMs, and MT with 3-state HMMs, respectively. Both of the MT methods used the global variance during HMM re-estimation. The confusion matrices list all the phonemes in the first column and show a mapping along each row of how each phoneme is recognized in the experiments. The matrices also list the number of insertions and deletions for each phoneme. After the deletions column, the percent correct recognition for each phoneme is listed. The next column lists the broad phonetic class of the phonemes, and the final column gives an average percent correct recognition for each broad phonetic class. Comparison between all three confusion matrices shows an improvement in some, but not all, of the phonemes with the MT method over the MFCC method.

	IY	EY	AE	IH	EH	AH	AX	ER	OW	AA	UH	UW	AO	V	Z	DH	ZH	SH	TH	F	S	HH	CH	JH	N	M	AW	OY	AY	R	L	Y	W	D	B	G	T	P	K	SIL	IX	EL	AXR	NG	EN	EM	Del	% Cor		Avg % Cor
IY	1123	123	13	44	20	19	12	24	18	8	24	79	3	84	5	9	44	10	5	4	13	32	13	19	17	63	3	7	9	7	3	259	26	17	25	40	16	31	7	0	64	4	59	65	3	0	220	45.4	VOWEL	
EY	37	419	9	19	7	4	0	4	2	0	4	5	1	4	2	3	6	1	2	0	4	28	6	5	4	0	5	3	5	9	0	10	3	3	7	9	1	18	3	0	27	3	8	22	1	0	44	58.8		
AE	3	127	865	60	118	47	15	9	11	10	19	9	7	39	7	12	4	5	12	4	10	88	14	7	22	19	81	5	36	7	4	27	10	3	13	20	12	58	8	0	76	3	39	34	4	3	218	43.6		
IH	95	132	26	490	78	60	28	30	44	3	62	76	1	89	18	40	36	15	24	15	14	67	15	11	55	47	9	26	7	12	4	102	15	27	37	49	14	87	14	0	111	8	62	56	10	1	458	22.1		
EH	7	49	90	37	366	19	9	25	30	9	16	4	6	20	5	5	2	4	9	7	3	39	6	2	7	14	30	26	21	12	4	11	9	6	15	13	8	32	5	0	25	6	28	15	3	0	179	34.6		
AH	39	53	69	156	119	572	53	94	192	61	111	75	27	255	29	50	31	25	39	31	31	72	21	11	81	85	67	71	68	26	35	42	97	33	95	51	29	145	25	0	135	30	49	30	31	6	760	16.6		
AX	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0		
ER	7	17	5	26	21	16	8	693	16	8	10	13	6	34	8	3	26	21	3	8	10	10	10	7	14	7	5	5	10	107	3	7	7	9	14	18	3	30	10	0	16	2	73	6	2	1	118	51.9		
OW	0	4	9	7	14	25	9	4	290	12	9	2	12	19	4	6	2	1	5	5	0	8	1	2	9	7	28	22	10	1	31	1	47	3	16	4	3	22	4	0	14	1	7	4	2	6	96	41.9		
AA	3	3	11	17	31	36	9	54	25	42	28	4	78	40	5	5	10	1 3	4	2	9	10	8	3	10	18	78	59	88	10	23	1	11	5	16	5	8	29	4	0	14	2	15	3	2	1	163	34.5		
UH	2	2	0	7	2	4	2	2	7	1	43	8	2	17	1	1	2	1	1	0	2	3	2	0	4	3	1	3	1	0	7	3	8	1	0	1	2	5	1	0	5	0	2	1	2	0	46	26.5		
UW	31	13	4	37	7	16	12	6	19	1	14	254	0	43	3	8	13	15	6	6	14	11	17	5	22	14	1	3	2	4	11	27	17	14	16	31	3	72	2	0	32	3	9	7	6	0	195	29.8		
AO	7	6	8	7	8	12	5	49	23	62	4	8	323	18	2	3	7	4	2	7	5	7	2	13	5	4	37	51	16	7	14	0	101	2	9	11	11	20	6	0	9	0	17	3	2	1	459	35.2		32.82%
V	1	4	2	2	4	2	8	1	12	3	9	4	1	413	3	8	7	0	12	27	1	6	4	7	9	5	5	6	1	5	0	3	13	4	26	4	2	15	4	0	12	1	9	2	6	0	65	61.4	CONSONANT	
Z	2	10	7	6 9	7	7	0	1	3	4	15	7	2	22	70	10	30	19	42	23	196	12	43	37	6	3	3	1	3	3	2	7	4	4	19	6	11	24	16	0	5	32	7	11	4	3	50	50.9		
DH	16	11	5	22	28	18	13	2	11	4	25	7	3	56	8	314	3	0	69	23	3	19	1	2	14	25	8	10	5	6	6	11	7	17	69	10	4	106	5	0	25	2	5	11	8	0	179	30.9		
ZH	1	0	0	3	0	1	1	0	0	0	0	1	0	1	4	0	38	3	0	0	1	3	5	3	0	0	1	0	0	0	0	2	0	1	1	0	1	3	0	0	0	0	0	0	0	0	2	51.4		
SH	2	1	3	5	0	5	2	2	0	0	2	1	0	6	9	5	15	515	2	30	23	5	54	5	1	0	0	1	1	1	1	11	1	0	2	0	3	5	5	0	3	0	0	1	0	0	31	70.7		
TH	2	4	3	1	2	2	0	2	2	0	5	0	0	2	2	9	1	2	100	11	4	0	1	0	1	0	0	2	0	1	0	1	3	2	4	3	1	22	2	0	3	0	4	1	0	0	10	48.8		
F	1	4	3	1	5	3	0	3	2	1	1	1	0	2	0	2	0	7	38	769	2	5	2	1	4	0	1	1	0	3	1	1	6	1	0	2	4	13	6	0	2	0	1	1	1	0	19	85.3		
S	3	7	11	2 12	8	11	1	6	8	1	7	4	2	22	19	18	12	79	60	26	1700	20	43	9	5	10	7	6	5	2	6	33	7	5	4	9	13	35	16	0	16	75	5	4	4	2	113	67.2		
HH	3	9	4	9	6	12	4	7	3	4	6	3	4	11	1	9	11	4	3	5	9	279	6	14	4	16	3	9	2	9	8	42	2	4	5	9	14	10	41	0	12	1	6	6	1	1	119	44.2		
CH	1	1	2	4	3	2	0	0	0	0	0	5	0	1	3	3	0	34	0	2	8	4	152	3	2	0	1	0	0	0	0	0	0	2	2	0	5	7	2	0	3	0	2	0	1	0	16	59.6		
JH	1	3	1	10	9	0	1	2	1	1	2	0	0	1	3	4	14	22	1	0	9	1	28	90	2	3	1	0	1	2	0	8	1	3	3	7	3	5	3	0	4	0	4	4	0	0	15	34.9		
N	20	25	28	27	68	40	45	15	31	18	31	32	23	103	15	25	11	9	17	13	12	42	12	9	1013	257	30	28	22	18	8	29	38	40	91	42	18	65	33	0	114	5	34	216	32	3	385	36.1		
M	3	14	13	9	13	12	17	12	17	6	7	12	5	39	1	27	3	2	5	3	5	16	6	4	60	835	10	7	5	4	4	8	18	17	84	21	7	20	7	0	36	2	8	41	20	3	122	56.9		51.94%
AW	1	1	8	1	2	0	2	5	6	12	1	0	5	6	2	3	0	0	2	0	0	3	2	0	1	5	105	2	7	2	3	0	2	1	4	1	2	7	1	0	5	0	2	0	0	1	22	49.3	diphthong	
OY	0	9	2	6	4	7	4	2	8	11	7	4	5	4	0	1	0	0	1	1	0	1	0	0	0	3	5	149	12	3	1	0	16	0	2	1	0	1	0	0	8	0	0	0	0	0	14	53.6		
AY	5	22	22	14	9	9	3	13	11	38	5	1	7	15	0	4	1	1	7	2	2	4	0	1	18	2	25	19	489	11	2	1	5	5	7	14	2	11	2	0	12	3	10	8	2	0	55	57.9		55.66%
R	12	35	28	28	26	21	12	500	16	23	57	13	37	66	8	18	9	38	9	19	7	26	38	21	17	37	23	32	38	1111	11	22	54	14	53	50	17	78	30	0	36	3	224	10	5	2	436	37.9	semivowel	
L	20	32	5	40	21	48	6	17	122	36	65	28	46	63	9	18	4	3	12	7	13	16	5	3	28	53	33	119	26	16	1115	14	200	6	27	16	16	46	12	0	46	8	32	16	5	2	282	45.1		
Y	29	10	4	13	5	1	0	2	4	2	2	1	1	7	1	1	8	7	1	0	1	7	5	10	5	2	0	3	0	4	2	534	6	3	6	5	1	6	0	0	6	0	9	7	1	0	66	74		
W	0	0	3	4	3	5	2	3	4	3	18	5	9	21	0	11	1	2	3	9	2	5	0	1	2	14	0	27	2	5	18	4	951	1	7	3	3	10	3	0	10	0	0	6	0	2	62	80.5		50.75%
D	14	46	17	49	96	16	16	17	15	11	33	24	2	66	9	46	16	20	17	12	22	59	26	73	33	58	13	11	11	6	7	22	14	443	188	48	33	137	22	0	37	4	25	25	2	0	344	23.8	consonant stop	
B	1	0	1	3	3	5	0	8	7	4	1	1	11	13	2	9	0	2	3	4	0	1	1	1	9	10	3	4	2	3	10	4	6	12	618	4	1	98	4	0	4	2	4	6	5	1	101	69.4		
G	8	8	10	9	11	44	3	12	4	3	8	4	2	42	0	19	2	5	0	4	2	30	3	6	11	22	4	3	3	6	2	22	6	18	54	353	4	29	33	0	5	3	7	17	1	1	62	41.9		
T	29	40	27	52	70	79	33	41	30	46	36	31	11	89	30	58	19	63	83	68	80	136	93	35	49	49	44	57	32	30	16	59	19	65	82	59	350	399	92	0	81	10	43	36	5	4	683	12.2		
P	0	2	2	1	3	3	0	1	4	1	4	1	2	10	1	6	0	0	14	19	2	16	3	3	5	2	4	1	1	1	0	2	4	3	17	1	4	904	16	0	2	0	0	0	1	0	24	84.8		
K	14	25	7	25	12	24	2	9	5	6	8	4	3	22	4	26	3	40	36	17	5	131	38	12	22	8	10	4	10	6	6	15	10	14	35	52	52	433	761	0	15	7	11	17	1	5	169	38.6		36.12%
SIL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3360	0	0	0	0	0	0	0			
IX	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	33.3		
EL	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
AXR	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
NG	1	10	2	0	5	3	2	2	3	2	1	1	2	6	2	0	1	3	2	2	0	2	1	2	13	20	2	1	2	3	2	2	5	4	16	5	4	4	1	0	13	0	0	207	1	0	23	57.5		
EN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
EM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
Sum	1544	1281	1329	1245	1216	1210	339	1680	1007	457	700	732	649	1771	295	799	392	982	651	1185	2224	1224	687	437	1584	1720	686	784	953	1463	1370	1347	1749	812	1690	977	685	3042	1206	3360	1044	220	820	899	174	49	6426			
Ins	175	363	171	388	361	354	84	172	193	160	291	123	166	535	108	246	105	216	244	345	167	428	137	110	286	226	1861	180	188	185	133	305	313	227	564	298	333	52	361	0	298	88	184	268	34	37				

Table 4. Confusion matrix for speech recognition results using MFCCs without delta and acceleration coefficients and 3 state HMMs.

	IY	EY	AE	IH	EH	AH	AX	ER	OW	AA	UH	UW	AO	V	Z	DH	ZH	SH	TH	F	S	HH	CH	JH	N	M	AW	OY	AY	R	L	Y	W	D	B	G	T	P	K	SIL	IX	EL	AXR	NG	EN	EM	Del	% Cor		Avg % Cor
IY	1482	16	41	4	3	0	0	12	62	27	83	60	6	10	6	71	3	8	6	0	13	65	131	4	14	96	59	131	7	1	4	6	3	21	19	8	3	96	48	0	0	0	2	8	4	0	50	56.1	VOWEL	
EY	20	282	11	2	2	0	0	4	27	5	14	22	0	5	1	17	0	3	3	1	6	25	37	7	4	56	25	61	4	1	3	0	2	12	3	2	1	32	25	0	0	0	0	0	2	0	30	38.8		
AE	44	15	165	11	5	1	0	2	55	19	61	61	0	15	1	68	0	4	9	5	22	65	119	9	15	55	49	117	7	4	4	14	7	29	17	13	0	97	54	0	1	0	0	0	0	0	65	49.8		
IH	144	57	83	41	7	3	0	2	143	53	103	100	2	33	13	87	0	11	17	5	43	138	257	18	26	211	158	265	7	5	6	11	5	47	32	11	8	221	119	0	5	0	0	0	0	0	183	1.6		
EH	68	32	59	2	34	0	0	1	58	30	39	44	0	15	1	39	0	1	2	4	18	65	121	4	11	122	88	105	0	2	0	1	1	30	8	3	1	106	56	0	2	0	0	0	0	0	65	2.9		
AH	179	82	122	14	14	10	0	4	219	96	150	165	1	70	7	162	0	5	18	8	73	236	473	26	51	313	308	400	6	22	2	21	12	87	55	5	2	327	197	0	5	0	1	1	1	0	257	0.3		
AX	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
ER	77	27	41	11	3	0	0	29	80	34	63	51	1	22	3	54	2	4	5	3	38	64	146	5	12	118	74	142	3	6	7	5	9	25	29	8	1	112	55	0	1	0	1	2	1	0	79	2.1		
OW	16	14	17	1	2	0	0	0	423	17	8	12	1	13	1	7	0	3	3	1	9	15	17	4	13	47	37	21	0	2	0	2	1	7	6	0	0	22	25	0	1	0	0	0	0	0	20	55.1		
AA	29	7	23	0	1	0	0	0	61	511	46	38	0	3	0	40	0	0	2	0	16	52	85	4	8	76	35	110	0	1	0	0	1	18	9	1	2	87	66	0	0	0	0	0	0	0	52	38.4		
UH	2	0	1	0	1	0	0	0	6	3	108	2	0	1	0	5	0	0	0	0	3	7	12	2	0	12	7	13	0	0	1	0	1	2	3	0	0	9	3	0	0	0	0	0	0	0	4	52.9		
UW	33	16	16	2	1	0	0	1	27	14	11	559	0	7	0	20	0	0	2	0	12	17	50	2	10	55	34	57	2	3	2	3	3	3	10	0	0	32	22	0	0	0	0	0	0	0	20	54.5		
AO	48	34	51	0	7	1	0	1	73	63	47	52	7	18	1	49	0	1	2	2	31	87	133	12	24	108	91	124	3	0	0	7	0	23	18	3	1	129	64	0	2	0	0	0	0	0	60	0.5		11.73%
V	32	14	13	1	2	0	0	0	48	13	38	27	1	83	0	32	0	0	1	0	14	28	76	3	2	66	25	82	0	1	0	1	1	13	9	2	0	51	34	0	1	0	0	0	0	0	24	11.6	CONSONANT	
Z	48	30	36	8	3	0	0	16	49	40	67	51	10	21	38	41	5	27	6	0	32	62	158	10	13	92	94	98	10	4	7	7	22	20	8	25	0	103	62	0	1	0	8	24	8	0	74	2.8		
DH	38	41	22	1	3	1	0	0	25	28	22	26	0	14	1	453	0	0	1	0	14	28	46	7	10	82	68	50	0	9	0	5	0	16	10	0	0	60	52	0	1	0	0	0	0	0	62	39.9		
ZH	2	2	1	0	0	0	0	1	0	0	1	5	2	1	0	1	1	2	1	0	3	4	10	1	1	1	2	6	0	0	1	0	1	1	1	3	0	10	3	0	0	0	1	2	0	0	5	1.4		
SH	29	56	17	0	3	0	0	2	25	12	13	25	0	4	0	26	0	2	0	0	63	23	164	8	1	61	21	64	0	2	1	2	0	18	6	0	0	37	36	0	1	0	0	1	0	0	36	0.3		
TH	8	4	12	0	1	0	0	0	7	4	5	11	0	2	0	7	0	0	5	0	5	8	35	2	1	24	17	18	0	0	0	0	0	5	3	0	0	15	9	0	1	0	0	0	0	0	6	2.4		
F	42	19	38	1	2	1	0	1	53	21	30	30	1	17	1	34	0	0	0	1	19	68	74	8	16	75	56	91	1	2	0	4	1	25	6	0	0	84	44	0	0	0	0	2	0	0	52	0.1		
S	30	37	38	6	2	1	0	11	54	23	70	34	5	27	3	41	4	11	0	0	1419	62	110	9	6	112	64	129	5	5	7	7	6	40	15	4	0	112	51	0	0	0	0	2	4	3	75	55.2		
HH	26	27	10	0	2	0	0	0	11	19	7	13	0	10	0	13	0	0	0	0	10	416	33	0	1	40	22	23	0	5	0	2	0	9	5	0	0	14	12	0	0	0	0	0	0	0	20	57		
CH	0	2	2	0	0	0	0	0	2	0	3	2	0	1	0	5	1	1	0	1	1	1	208	0	1	6	3	5	0	0	1	2	0	1	4	0	0	4	4	0	0	0	0	1	1	0	8	79.1		
JH	15	3	7	2	0	0	0	2	12	5	13	6	2	4	2	9	1	0	1	2	4	13	15	25	1	19	13	14	1	1	3	2	1	7	7	4	0	30	10	0	0	0	1	1	1	0	14	9.7		
N	111	62	110	12	8	1	0	7	186	55	126	135	4	25	3	118	3	12	5	3	51	132	280	18	325	272	144	256	9	11	7	15	9	67	32	12	1	249	167	0	6	0	1	7	1	1	133	10.6		
M	19	11	6	1	3	0	0	3	21	10	21	20	0	7	0	22	2	2	1	0	7	35	49	1	4	1191	11	36	1	2	2	4	2	7	10	1	0	37	15	0	1	0	0	4	0	0	21	75.9		30.79%
AW	2	1	0	2	0	0	0	0	1	3	6	3	0	0	1	2	0	0	2	0	5	3	6	1	0	4	158	9	0	0	0	0	0	3	6	0	0	10	4	0	0	0	0	0	0	0	3	68.1	diphthong	
OY	2	2	2	0	0	0	0	0	0	4	0	4	0	3	0	0	0	0	0	0	1	0	2	2	0	4	5	254	0	1	0	0	0	0	0	0	0	2	1	0	0	0	0	0	0	0	3	87.9		
AY	32	28	37	2	0	0	0	0	57	29	33	29	0	17	4	33	0	1	1	0	13	32	71	3	6	107	65	107	8	6	0	5	2	16	7	5	1	74	34	0	0	0	0	0	0	0	34	0.9		30.30%
R	95	87	104	7	7	0	0	22	96	55	99	111	7	40	16	96	12	14	5	2	73	144	311	26	17	272	184	286	11	470	9	8	21	52	29	24	0	238	139	0	2	0	8	22	4	1	144	14.6	semivowel	
L	96	74	94	7	10	0	1	10	156	64	103	84	7	36	8	109	0	11	3	3	84	134	262	25	19	268	159	260	10	13	34	6	10	53	34	18	2	190	131	0	5	0	2	5	3	0	154	1.3		
Y	28	7	21	1	5	1	0	0	19	7	54	16	0	6	0	24	0	1	2	5	17	47	75	2	4	60	33	58	0	1	0	65	0	12	17	7	2	92	38	0	2	0	0	0	0	0	59	8.9		
W	54	24	39	0	3	0	0	0	73	29	44	38	0	26	2	62	0	0	1	0	30	63	94	13	12	107	115	124	0	7	0	2	6	18	19	1	0	97	51	0	2	0	0	0	0	0	88	0.5		7.45%
D	95	75	78	3	5	0	0	6	48	45	64	44	4	43	3	60	6	7	4	0	40	65	143	19	19	149	129	140	5	4	3	12	7	583	8	9	1	112	55	0	3	0	2	6	3	1	97	27.7	consonant stop	
B	35	28	30	0	1	0	0	0	38	20	29	33	0	12	0	34	0	0	0	0	24	43	86	12	4	91	76	83	0	3	0	1	0	12	120	0	2	85	42	0	2	0	0	0	0	0	46	12.7		
G	38	25	43	1	2	0	0	2	34	15	41	43	0	10	0	34	0	0	0	0	14	38	69	6	6	90	59	99	1	2	0	7	1	22	6	10	0	89	54	0	2	0	0	1	1	0	40	1.2		
T	153	123	92	10	11	0	0	36	135	74	146	93	14	53	14	93	10	23	8	4	76	178	416	35	28	273	201	293	19	21	16	15	40	72	27	21	22	335	135	0	1	1	6	17	6	2	195	0.7		
P	16	11	6	0	1	0	0	1	4	5	7	7	1	4	2	3	1	1	1	1	3	4	22	3	3	13	26	17	0	0	0	0	1	3	1	1	0	889	11	0	0	0	0	1	0	0	20	83.1		
K	59	36	41	0	2	1	0	3	49	28	46	43	0	21	1	28	0	1	4	0	14	55	135	6	9	56	46	100	3	12	2	5	0	27	4	5	0	104	1152	0	2	0	0	1	3	0	37	54.8		26.59%
SIL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3360	0	0	0	0	0	0	0			
IX	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
EL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
AXR	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
NG	11	7	7	1	1	0	0	4	7	8	14	16	0	11	1	22	2	3	0	0	7	19	29	3	4	29	25	33	1	2	2	3	8	12	5	3	0	36	26	0	1	0	1	4	1	0	14	1.1		
EN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
EM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
Sum	3259	1419	1536	154	157	21	1	183	2444	1488	1836	2116	76	710	134	2022	53	159	121	51	2328	2541	4560	345	701	4833	2786	4282	124	631	124	250	184	1418	608	209	50	4429	3106	3360	51	1	34	112	44	8	2349			
Ins	9001	826	7612	34	158	55	0	30	796	51742	412	2392	19	3021	21	1753	13	28	33	23	217	32362	6313	502	30	920	40795	5782	27	2672	23	309	31	712	478	26	819	2331	2991	0	256	0	7	25	5	1				

Table 5. Confusion matrix for speech recognition results using MT data and 1-state HMMs.

	IY	EY	AE	IH	EH	AH	ER	OW	AA	UH	UW	AO	V	Z	DH	ZH	SH	TH	F	S	HH	CH	JH	N	M	AW	OY	AY	R	L	Y	W	D	B	G	T	P	K	SIL	EL	AXR	NG	EN	EM	Del	% Cor		Avg % Cor
IY	678	57	32	0	17	1	8	10	24	81	31	9	66	2	3	16	11	7	3	28	82	117	2	37	22	29	85	5	10	1	2	6	0	17	8	0	183	1	0	0	0	0	1	0	1001	40.1	VOWEL	
EY	14	174	20	0	4	0	2	5	2	24	6	1	22	1	0	3	1	1	0	10	10	21	1	19	7	16	32	0	4	0	2	0	0	7	2	0	50	0	0	0	0	0	0	0	296	37.7		
AE	22	36	739	0	25	2	0	5	8	53	8	0	57	0	4	0	0	4	6	17	28	34	5	33	19	45	90	0	15	0	0	0	0	9	8	0	152	2	0	0	0	0	0	0	778	51.8		
IH	55	46	58	1	23	3	0	12	22	95	21	0	93	0	9	0	0	12	8	39	62	80	14	35	24	67	133	0	19	0	0	0	1	18	16	0	181	2	0	0	0	0	0	0	1531	0.1		
EH	13	15	69	0	112	0	0	7	10	40	10	0	46	0	5	0	0	3	5	13	29	34	4	11	12	34	74	0	9	0	0	0	0	5	13	0	64	0	0	0	0	0	0	0	601	17.6		
AH	59	66	100	0	31	38	1	34	52	173	24	0	168	2	8	0	2	9	18	62	122	132	12	40	44	115	233	0	30	0	0	1	5	24	25	0	303	6	0	0	1	0	0	0	2267	2		
AX	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0		
ER	11	7	28	0	7	0	62	6	15	64	11	1	49	5	3	7	1	1	3	15	32	48	4	21	8	33	110	0	12	2	1	3	3	13	6	0	128	2	0	0	1	0	0	0	730	8.6		
OW	9	12	14	0	7	1	1	90	9	18	5	1	24	1	2	2	1	0	2	9	17	13	2	6	6	52	49	0	9	0	3	0	1	6	7	0	43	1	0	0	0	0	0	0	365	21.3		
AA	8	10	28	0	4	0	0	7	240	38	5	0	39	0	3	0	0	1	3	20	17	38	2	14	6	70	79	0	6	0	0	0	1	4	5	0	100	2	0	0	0	0	0	0	634	32		
UH	3	3	0	0	1	0	0	2	2	75	0	0	8	0	0	0	0	1	0	4	2	5	0	5	2	3	13	0	0	0	0	0	0	2	0	0	10	0	0	0	0	0	0	0	67	53.2		
UW	12	15	19	0	6	0	1	6	12	28	180	1	32	0	0	2	2	1	3	10	19	20	4	22	5	17	38	0	8	0	0	2	0	3	6	0	88	1	0	0	0	0	0	0	483	32		
AO	19	26	39	0	6	1	1	8	30	37	21	34	44	0	1	0	0	2	3	10	25	34	4	27	9	26	76	0	3	0	0	0	1	20	8	0	131	1	0	0	0	0	0	0	730	5.3		22.96%
V	3	6	6	0	3	0	0	1	3	25	4	0	301	0	0	1	0	2	0	5	10	15	2	8	5	4	22	0	2	0	0	0	0	7	2	0	37	2	0	0	0	0	0	0	262	63.2	CONSONANT	
Z	13	9	33	1	5	2	16	6	8	63	12	16	46	18	1	28	19	1	6	55	41	81	4	17	9	27	79	13	8	3	7	15	1	9	4	0	91	2	0	0	8	5	3	0	653	2.3		
DH	10	18	18	0	9	2	0	7	11	38	7	0	42	0	31	0	0	2	1	8	23	20	4	18	8	42	75	0	13	0	0	0	3	12	3	0	107	3	0	0	0	0	0	0	661	5.8		
ZH	0	1	1	0	0	0	0	1	0	3	2	0	2	0	0	5	3	0	0	1	1	4	0	1	0	0	4	0	0	0	0	1	0	0	1	0	8	0	0	0	1	0	0	0	36	12.5		
SH	4	14	6	0	0	0	0	2	2	10	3	0	17	0	1	1	14	1	2	20	19	150	1	8	7	7	21	1	5	0	0	0	1	4	4	0	29	2	0	0	0	0	0	0	403	3.9		
TH	3	2	6	0	4	0	0	0	3	9	2	0	8	0	0	0	0	5	1	3	7	9	0	1	4	6	11	0	1	0	1	0	0	1	2	0	20	0	0	0	0	1	0	0	105	4.5		
F	5	4	9	0	5	1	1	5	6	58	7	2	29	1	1	2	0	3	22	21	25	41	2	13	5	18	52	0	3	0	1	0	2	3	3	0	75	6	0	0	0	0	0	0	489	5.1		
S	7	14	14	1	8	0	9	3	8	60	16	5	50	2	3	9	1	0	5	1216	28	89	5	17	18	36	79	6	11	3	5	10	0	10	10	0	127	2	0	0	1	3	0	0	753	64.3		
HH	29	8	11	0	2	1	0	3	8	17	6	0	12	0	1	0	0	0	2	9	209	34	0	4	8	11	27	0	1	0	0	0	0	4	5	0	35	2	0	0	0	0	0	0	301	46.5		
CH	2	0	1	0	0	0	0	2	0	5	1	0	2	0	0	2	0	0	1	4	5	119	1	2	4	1	5	2	1	0	0	0	0	2	0	0	18	0	0	0	0	0	0	0	91	66.1		
JH	2	5	2	0	1	0	1	1	0	12	4	1	11	1	0	4	1	1	0	2	5	18	6	3	0	7	15	2	1	0	2	1	0	2	1	0	29	0	0	0	1	2	1	0	128	4.1		
N	25	33	38	0	12	1	8	12	22	106	32	4	92	7	8	11	3	5	2	40	56	56	8	743	39	58	145	1	22	1	2	2	1	19	18	0	245	1	0	0	1	3	0	0	1310	39.5		
M	9	22	25	0	15	0	3	7	6	72	15	2	59	0	7	8	4	2	3	17	25	42	1	21	265	31	102	1	9	0	1	4	3	4	10	1	132	0	0	0	0	2	0	0	660	28.5		35.98%
AW	1	1	3	0	1	0	0	1	4	12	0	0	3	0	0	0	0	0	1	3	6	4	0	2	3	81	10	1	0	0	0	0	0	0	1	0	23	0	0	0	0	0	0	0	74	50.3	diphthong	
OY	0	3	1	0	1	0	0	1	2	2	0	0	0	0	0	0	0	0	0	2	2	2	0	1	1	5	214	0	1	0	0	0	0	1	0	0	7	0	0	0	0	0	0	0	46	87		
AY	6	13	31	0	13	1	0	2	17	33	7	0	39	1	2	0	0	2	6	18	15	21	1	23	8	37	60	12	10	0	0	0	1	8	6	0	58	1	0	0	0	0	1	0	446	2.6		35.70%
R	19	28	53	0	13	2	4	8	22	78	56	0	136	10	11	30	3	3	7	43	41	99	6	46	20	63	231	11	561	0	1	6	5	28	16	0	261	10	0	0	8	8	3	0	1420	28.8	semivowel	
L	21	26	42	0	26	4	6	27	35	115	18	7	118	0	7	8	5	5	8	37	51	84	1	35	29	83	195	3	17	16	3	9	1	25	8	0	198	6	0	0	3	1	2	0	1472	1.2		
Y	23	14	8	0	4	0	0	3	3	20	10	0	22	0	0	0	0	1	2	11	24	40	4	18	6	14	42	0	6	0	11	0	1	4	2	0	62	0	0	0	0	0	0	0	433	3.1		
W	4	10	23	0	5	3	0	5	11	30	7	0	38	0	3	0	0	2	2	9	17	24	2	29	17	36	95	0	1	0	0	29	2	17	6	0	97	0	0	0	0	0	0	0	720	5.5		15.00%
D	22	17	30	1	19	0	4	9	31	89	15	4	81	2	3	15	10	4	7	27	49	46	6	29	18	47	83	3	12	1	3	3	28	10	9	0	224	2	0	0	1	0	2	1	1238	2.9	consonant stop	
B	10	6	17	0	8	0	0	6	10	21	4	0	39	0	0	0	0	2	2	10	14	16	0	10	8	31	44	0	5	0	0	1	1	107	2	0	113	0	0	0	0	0	0	0	505	22		
G	5	7	14	0	6	2	2	3	7	33	12	1	30	0	5	0	0	2	2	13	21	14	2	14	6	26	35	0	1	0	0	0	1	9	84	0	113	3	0	0	0	1	1	0	430	17.7		
T	34	46	60	0	19	1	32	13	27	113	35	21	96	11	6	27	15	7	6	34	80	147	3	50	39	85	145	4	18	2	19	30	0	13	15	0	365	6	0	1	6	8	4	0	1900	0		
P	3	2	5	0	1	0	2	0	3	7	4	1	8	3	0	1	0	1	0	5	5	9	0	3	3	6	8	1	0	0	0	0	0	1	0	0	809	4	0	0	0	0	0	0	195	90.4		
K	25	22	26	0	16	2	4	8	10	87	18	0	71	2	3	2	1	0	3	30	67	67	4	55	14	45	84	3	12	0	1	4	3	16	6	0	292	46	0	0	0	1	2	1	1088	4.4		19.46%
SIL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3360	0	0	0	0	0	0			
IX	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	2	0		
EL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
AXR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1			
NG	1	4	6	0	3	0	2	2	4	15	5	1	7	3	1	3	1	1	0	3	11	7	1	11	5	8	20	5	1	1	0	5	1	8	0	0	33	2	0	0	1	1	1	0	199	0.5		
EN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
EM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
Sum	1189	802	1635	4	442	68	170	330	689	1859	624	112	2007	72	132	187	98	94	145	1884	1302	1834	118	1452	713	1322	2916	74	847	30	65	132	67	452	323	1	5042	118	3360	1	33	36	21	2	25504			
Ins	46	76	79	1	40	6	2	17	39	150	32	6	238	2	7	5	5	10	10	55	92	170	8	77	40	108	207	1	29	0	2	0	0	21	20	0	268	6	0	0	1	4	0	0				

Table 6. Confusion matrix for speech recognition results using MT data and 3-state HMMs.

Table 7 lists the phonemes with improved percent correct recognition performance using the MT features with either 1-state HMMs or 3-state HMMs over the MFCC features (without delta and acceleration coefficients). Each of the MT methods used the global variance for all re-estimations. Both the percent correct recognition rates and the percent accuracy rates are shown for each type of feature set. In some cases the accuracy percentage was negative. Obviously, the MT data does not perform equally well in all 1-state and 3-state cases. Even though the MT features did not outperform the MFCCs overall, improvement was found for some phonemes. The increase in performance found for these phonemes with the MT might be exploited by fusing the particular phoneme models that have improved performance for the MT with existing models of HMMs trained with MFCCs. Also, it is important to note that the MFCCs

	MFCC		**Mellin-1 State**		**Mellin-3 State**	
Phoneme	% Cor	% Acc	% Cor	% Acc	% Cor	% Acc
IY	45.41%	38.33%	56.07%	-284.49%	40.07%	37.35%
AE	43.55%	34.94%	13.32%	-601.05%	51.82%	46.28%
OW	41.91%	14.02%	55.08%	-48.57%	21.28%	17.26%
AA	5.01%	-14.06%	38.36%	-3846.17%	32.00%	26.80%
UH	26.54%	-153.09%	52.94%	-149.02%	53.19%	-53.19%
UW	29.85%	15.39%	54.48%	-178.65%	31.97%	26.29%
V	61.37%	-18.13%	11.62%	-411.48%	63.24%	13.24%
DH	30.88%	6.69%	39.95%	-114.64%	5.79%	4.49%
HH	44.22%	-23.61%	56.99%	-4376.16%	46.55%	26.06%
CH	59.61%	5.88%	79.09%	-2321.29%	66.11%	-28.33%
N	36.09%	25.90%	10.62%	9.64%	39.48%	35.39%
M	56.88%	41.49%	75.91%	17.27%	28.49%	24.19%
AW	49.30%	-824.41%	68.10%	-17515.95%	50.31%	-16.77%
OY	53.60%	-11.15%	87.89%	-1912.80%	86.99%	2.85%
D	23.80%	11.61%	27.66%	-6.12%	2.90%	2.90%
P	84.80%	79.92%	83.08%	-134.77%	90.39%	60.45%
K	38.59%	20.28%	54.75%	-87.40%	4.37%	3.80%

Table 7. The results of the HTK recognition process for phonemes with improved recognition performance using MT features over MFCC features.

have been finely tuned over decades of research, while the MT features are just starting to be investigated, and thus further research on the MT features may yield better results.

V. Discussion and Recommendations

Research previously conducted on vowel recognition using the MT [11] [17] suggested the possibility of improved speech recognition results with MT features. This previous research used a Mixture of Gaussians model on single simulated vowel frames to perform vowel recognition [3] [11]. Since all broad phonetic classes of real speech were used in this research, not just synthetic vowels, it is more general than previous research.

This research reported here differs from this previous research in that it uses HMMs to conduct the phoneme recognition experiments. Since the HMMs are based on having synchronous feature vectors, adding the extra step of SAI synchronization was necessary. This step, while converting the SAI to synchronous form, might have contributed to a decrease in recognition performance results due to some of the design choices made. Investigating the tradeoff of some of these design choices related to synchronization may lead to better recognition performance. Alternatively, a method of speech recognition that does not require synchronous data (thus enabling the SAI to remain asynchronous) may produce better results.

There are a number of variations to try in future research. One variation to try is feature normalization. The MT features were not normalized to take into account overall magnitude differences between frames utterances. Feature mean and variance normalization, for example, may improve results. Another variation to try is using more than one mixture for each state in the HMM models. This would make the models more complicated but might make them more discriminating, thereby leading to improved results. Also, changing the algorithms used to calculate the various components of the

AIM to more complex ones may yield a tradeoff of computational complexity for improved results.

Rather than increasing the complexity of the recognition system, methods that reduce the complexity of the system but that retain as much information as possible in the data may be beneficial. For instance, principal components analysis (PCA) is a linear transformation that reduces the dimensionality of a dataset while retaining as much of the information as possible contained in the dataset. Another possible method for reducing complexity is to use marginal distributions (*e.g.,* temporal and spectral profiles), of the image data, where feature vectors that sum down the rows and across the columns of the data matrix are employed. Figure 18 shows an example of the marginal distributions of a SAI image frame. Similar processing could be done with the MT images. While PCA and marginal distributions might reduce feature information, they also might allow better trained HMMs given the small amount of data available in the TIMIT database.

As shown in the results section, some phonemes did increase in correct recognition performance when MT features were used instead of MFCC features. It may be beneficial to attempt fusion of the phoneme models trained with MT data that show improvement with existing MFCC-based models that perform relatively well.

In summary, using the MT features lead to improved phoneme recognition rates for some phonemes. The results obtained here further the AFRL/HECP mission in improving human-machine collaboration, and future research should be able to use these results to explore ways to further improve speech recognition.

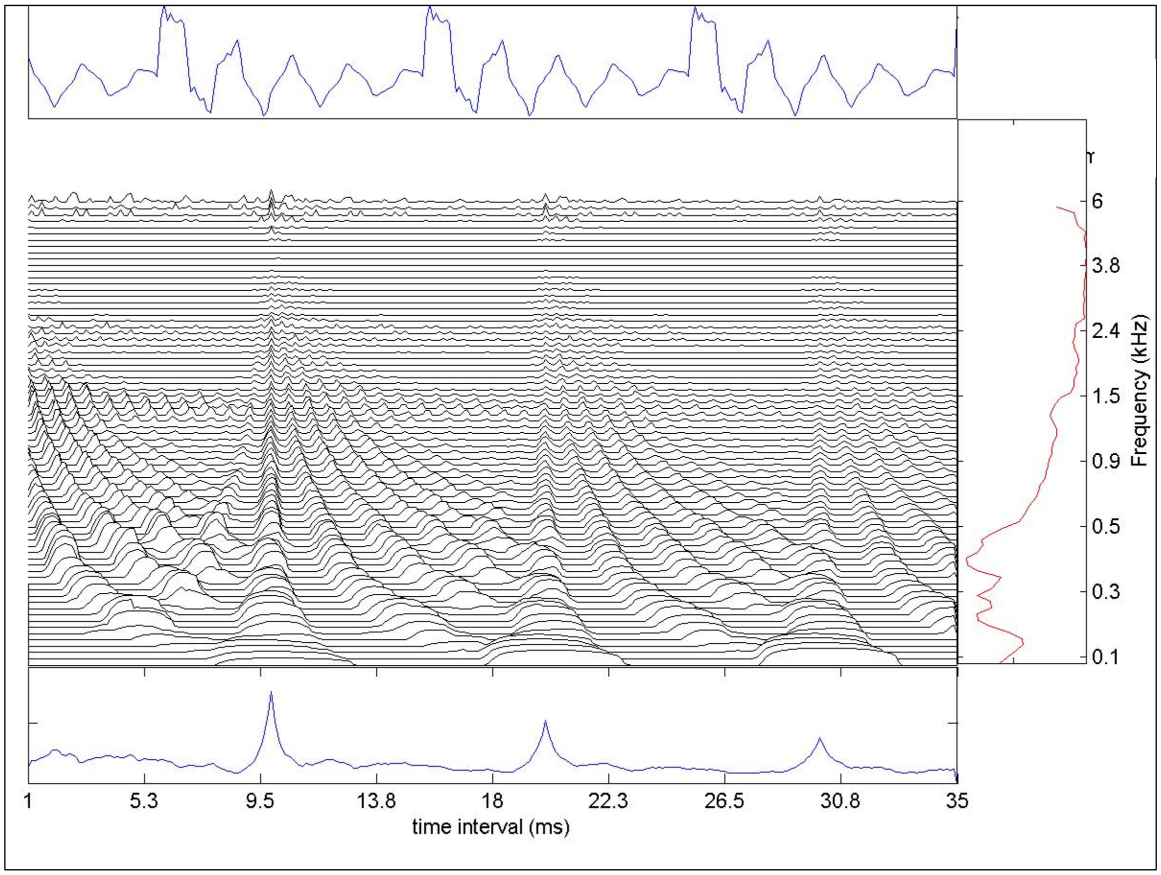

Figure 18. Marginal distributions of a SAI frame. The waveform at the top of the figure is the original speech waveform. The waveform at the bottom is the temporal profile of the SAI image, which is obtained by summing down the columns of the SAI data matrix. The waveform on the right is the spectral profile of the SAI image, which is obtained by summing across the rows of the SAI data matrix.

Appendix A: All Parameters Matlab Script

```
% Parameters
% for the project:
%%%%%%%%%%%%

%%%%%%%%%%%%
% Signaloptions
% all_options.signal.signal_filename='';
% all_options.signal.start_time=0;
% all_options.signal.duration=;
% all_options.signal.samplerate=16000;
% all_options.signal.original_start_time=0;
% all_options.signal.original_duration=;
% all_options.signal.original_samplerate=16000;


%%%%%%%%%%%%
% outer/middle ear filter function
all_options.pcp.none.generatingfunction='gennopcp';
all_options.pcp.none.displayname='no correction by outer/middle ear';
all_options.pcp.none.revision='$Revision: 1.2 $';


%%%%%%%%%%%%
% bmm
all_options.bmm.gtfb.generatingfunction='gen_gtfb';
all_options.bmm.gtfb.displayname='Gamma tone filter bank';
all_options.bmm.gtfb.revision='$Revision: 1.6 $';
all_options.bmm.gtfb.nr_channels=35;
all_options.bmm.gtfb.lowest_frequency=100;
all_options.bmm.gtfb.highest_frequency=6000;
all_options.bmm.gtfb.do_phase_alignment='off';
all_options.bmm.gtfb.phase_alignment_nr_cycles=3;
all_options.bmm.gtfb.b=1.019;


%%%%%%%%%%%%
% nap
all_options.nap.irinonap.generatingfunction='gen_irinonap';
all_options.nap.irinonap.displayname='Irinos nap';
all_options.nap.irinonap.revision='$Revision: 1.1 $';


%%%%%%%%%%%%
% strobes
all_options.strobes.irinostrobes.generatingfunction='genirinostrobes';
all_options.strobes.irinostrobes.displayname='Irinos Strobes';
all_options.strobes.irinostrobes.revision='$Revision: 1.2 $';

%%%%%%%%%%%%
```

```
% sai
all_options.sai.irinosai.generatingfunction='genirinosai';
all_options.sai.irinosai.displayname='time integration stabilized
auditory image';
all_options.sai.irinosai.revision='$Revision: 1.1 $';
all_options.sai.irinosai.start_time=0;
all_options.sai.irinosai.maxdelay=0.035;
all_options.sai.irinosai.buffer_memory_decay=0.0003;
all_options.sai.irinosai.frames_per_second=200;
all_options.sai.irinosai.delay_time_strobe_weight_decay=0.0002;


%%%%%%%%%%%%%
% user defined module
all_options.usermodule.mellin.generatingfunction='gen_mellin';
all_options.usermodule.mellin.displayname='mellin Image';
all_options.usermodule.mellin.displayfunction='displaymellin';
all_options.usermodule.mellin.revision='$Revision: 1.12 $';
all_options.usermodule.mellin.do_all_frames=1;
all_options.usermodule.mellin.framerange=[ 0 0];
all_options.usermodule.mellin.do_all_image=1;
all_options.usermodule.mellin.audiorange=[ 1 200];
all_options.usermodule.mellin.flipimage=0;
all_options.usermodule.mellin.ssi=0;
all_options.usermodule.mellin.log=0;


%%%%%%%%%%%%%
% graphics
all_options.graphics.ams.minimum_time=0.001;
all_options.graphics.ams.maximum_time=0.035;
all_options.graphics.ams.is_log=1;
all_options.graphics.ams.time_reversed=1;
all_options.graphics.ams.display_time=0;
all_options.graphics.ams.time_profile_scale=1;
all_options.graphics.autocorr.minimum_time=0;
all_options.graphics.autocorr.maximum_time=0.035;
all_options.graphics.autocorr.is_log=0;
all_options.graphics.autocorr.time_reversed=0;
all_options.graphics.autocorr.display_time=0;
all_options.graphics.autocorr.time_profile_scale=1;
all_options.graphics.grouped.minimum_time=0.001;
all_options.graphics.grouped.maximum_time=0.035;
all_options.graphics.grouped.is_log=1;
all_options.graphics.grouped.time_reversed=1;
all_options.graphics.grouped.display_time=0;
all_options.graphics.grouped.time_profile_scale=1;
all_options.graphics.grouped.plotstyle='waterfall';
all_options.graphics.grouped.colormap='summer';
all_options.graphics.grouped.colorbar='off';
all_options.graphics.grouped.viewpoint=[ 0 80];
all_options.graphics.grouped.camlight=[ 50 0; 30 90];
all_options.graphics.grouped.lighting='phong';
all_options.graphics.grouped.shiftcolormap=0.8;
all_options.graphics.gtfb.is_log=0;
all_options.graphics.gtfb.time_reversed=0;
all_options.graphics.gtfb.plotstyle='waterfall';
all_options.graphics.gtfb.colormap='cool';
all_options.graphics.gtfb.colorbar='vertical';
all_options.graphics.gtfb.camlight='left';
all_options.graphics.gtfb.lighting='phong';
all_options.graphics.hcl.is_log=0;
all_options.graphics.hcl.time_reversed=0;
all_options.graphics.hcl.plotstyle='waterfall';
all_options.graphics.hcl.colormap='hot';
all_options.graphics.hcl.colorbar='off';
all_options.graphics.irinosai.minimum_time=0.001;
all_options.graphics.irinosai.maximum_time=0.035;
all_options.graphics.irinosai.is_log=1;
all_options.graphics.irinosai.time_reversed=1;
all_options.graphics.irinosai.display_time=0;
all_options.graphics.irinosai.time_profile_scale=1;
all_options.graphics.irinosai.plotstyle='waterfall';
all_options.graphics.irinosai.colormap='summer';
all_options.graphics.irinosai.colorbar='off';
all_options.graphics.irinosai.viewpoint=[ 0 80];
all_options.graphics.irinosai.camlight=[ 50 0; 30 90];
all_options.graphics.irinosai.lighting='phong';
all_options.graphics.irinosai.shiftcolormap=0.8;
all_options.graphics.mellin.is_log=0;
all_options.graphics.mellin.time_profile_scale=100;
all_options.graphics.scaleprofile.is_log=0;
all_options.graphics.scaleprofile.time_profile_scale=1;
all_options.graphics.scaleprofile.frequency_profile_scale=1;
all_options.graphics.scaleprofile.minimum_time_interval=0.5;
all_options.graphics.scaleprofile.maximum_time_interval=20.5;
all_options.graphics.ti1992.minimum_time=0.001;
all_options.graphics.ti1992.maximum_time=0.035;
all_options.graphics.ti1992.is_log=1;
all_options.graphics.ti1992.time_reversed=1;
all_options.graphics.ti1992.display_time=0;
all_options.graphics.ti1992.time_profile_scale=1;
all_options.graphics.ti1992.plotstyle='waterfall';
all_options.graphics.ti1992.colormap='pink';
all_options.graphics.ti1992.colorbar='off';
all_options.graphics.ti2003.minimum_time=0.001;
all_options.graphics.ti2003.maximum_time=0.035;
all_options.graphics.ti2003.is_log=1;
all_options.graphics.ti2003.time_reversed=1;
all_options.graphics.ti2003.display_time=0;
all_options.graphics.ti2003.time_profile_scale=1;
all_options.graphics.ti2003.plotstyle='waterfall';
all_options.graphics.ti2003.colormap='summer';
all_options.graphics.ti2003.colorbar='off';
all_options.graphics.ti2003.viewpoint=[ 0 80];
all_options.graphics.ti2003.camlight=[ 50 0; 30 90];
all_options.graphics.ti2003.lighting='phong';
all_options.graphics.ti2003.shiftcolormap=0.8;
```

Appendix B: Wave File Conversion Perl Script

```
#!/usr/bin/perl

# Read in a TIMIT .wav - strip off SPHERE header and convert to
MS WAV format

$list_in = "filelist.txt"; # reads in the file list of TIMIT
files

open(IN, "$list_in");
 chop(@in = <IN>);
close(IN);
$cnt = $#in;

for($x=0; $x<=$cnt; $x++) {
 ($root,$ext) = split('\.', $in[$x]);
 print "$root\n";
 `/tools/NIST/bin/h_strip ${root}.wav ${root}.raw`;
 `mv ${root}.wav ${root}.wav.orig`;
 `sox -r 16000 -w -s ${root}.raw ${root}.wav`;
}
```

Appendix C: Main Program Matlab Code

```
% Main Program
% 2LT Jesse Hornback
% EENG 799
% 27 July 05

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% initialization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
echo off
clear;clear global;clc
global result
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% choose whether to compute:
% 1) SAI only, 2) Mellin only, 3) SAI and Mellin, 4) HTK format
z = 3;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% decide if you want plots: 0) off, 1) on
plotting = 1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% if plotting, decide whether to plot a portion (frames == 0) or
frames at
% a time (frames == 1)
frames = 0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% to see code in command window
echo on
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% load parameters here
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
echo off

[files,num_files] = file_list;num_files = 1;
path1 = 'M:/timit/mellin/'; % path for the wave files
%path1 = 'C:/MATLAB6p5/aim2003/Sounds/Tones and Misc';
path2 = 'H:/timit/mellin/'; % path for the SAI files
path3 = 'H:/timit/mellin/'; % path for the Mellin files
path4 = 'H:/timit/mellin/'; % path for the HTK files

% load the parameter m file to get the all_options struct array
current_directory = pwd;
cd ../parameter_files
run all_parameters
cd(current_directory)

%%%%%%%%%%%%%%%%%%%%
```

```
if z ~= 4


for i = 1:num_files
fprintf('\n << iteration %-0.0f >>\n\n',i)

tic
[signal_filename,dir] = get_parameter_file(i,files);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% option to adjust the Mellin transform "resolution"
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
all_options.usermodule.mellin.c_2pi=linspace(0,30,180); %
original length = 601 (tried 180,50)
all_options.usermodule.mellin.TFval=linspace(0,16,45); % original
length = 321 (tried 45,10)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% load some variables that will be needed later
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
abridge_sai = 1;
all_options.signal.signal_filename = signal_filename;
all_options.signal.start_time=0;
max_delay = all_options.sai.irinosai.maxdelay;
c_2pi = all_options.usermodule.mellin.c_2pi;
% load the wave file
current_directory = pwd;
cd(fullfile(path1,'MSwav_files'));
[y,Fs,nbits,opts] = wavread(signal_filename);
cd(current_directory);
all_options.signal.duration = length(y)/Fs;clear y
signal_duration = all_options.signal.duration;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%% start computations for differnt cases (1-3)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if z == 1

echo on
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% compute SAI
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
echo off

% call the aim_ng2_sai (modified) function
[result,strobes,newSigLen] =
aim_ng2_sai(all_options,signal_duration,max_delay,path1,abridge_s
ai);

% save the SAI to a .mat file for use later
save_sai(result.data.sai,i,signal_filename,dir,path2)


time(i) = toc;

if plotting == 0
    clear result
end

end

if z == 2

echo on
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% compute Mellin transform
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
echo off

[result] =
aim_ng2_mellin(all_options,signal_duration,signal_filename,max_de
lay,i,dir,path1,path2);

% save the MT to a .mat file for use later
save_mellin(result.data.usermodule,i,signal_filename,dir,path3)


time(i) = toc;

if plotting == 0
    clear result
end

end

if z == 3

echo on
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% compute SAI and Mellin transform
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
echo off
```

```
% call the aim_ng2 (modified) function
[result,strobes,newSigLen] =
aim_ng2(all_options,signal_duration,max_delay,path1,abridge_sai);


%%%%%%%%%%%%%%
% the SAI and Mellin data are not saved to a .mat file because
they are
% calculated on the fly
%%%%%%%%%%%%%%


time(i) = toc;

if plotting == 0
    clear result
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% now collect the SAI and Mellin data from the result struct
array and plot
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

warning off

if plotting

% specify what part of SAI image to plot and Mellin will be
calculated in x2
x1 = 1200; % x-axis from 1:x
%y = 200;    % y-axis from y:end

if z == 1

% call the harvest_sai_data function
sai_image = harvest_sai_data(result);

clear result

% plot part of the SAI
plot_sai_image(sai_image,signal_duration,x1) % use this if doing
only the SAI

end

if z == 2

% call the harvest_mellin_data function
mellin_image = harvest_mellin_data(result);
```

```
clear result


% scale x1 value to Mellin image so that the portion of the
Mellin image
% displayed corresponds to the portion of the SAI image displayed
x2 = x1*2/3;

% plot part of the Mellin
plot_mellin_mesh(mellin_image,signal_duration,c_2pi,x2)

end

if z == 3


if frames == 0

% call the harvest_sai_data function
sai_image = harvest_sai_data(result);

% call the harvest_mellin_data function
mellin_image = harvest_mellin_data(result,frames);

clear result

% scale x1 value to Mellin image so that the portion of the
Mellin image
% displayed corresponds to the portion of the SAI image displayed
x2 = x1*(size(mellin_image,2)/size(sai_image,2));

%%% plot part of the SAI and Mellin %%%
plot_sai_mellin_image(sai_image,mellin_image,signal_duration,c_2p
i,x1,x2)
plot_mellin_mesh(mellin_image,signal_duration,c_2pi,x2)

end


if frames == 1

%%% plot frames at a time %%%
%%plot_sai_mellin_image_frames_obsolete(sai_image,mellin_image,si
gnal_duration,c_2pi,newSigLen)
plot_sai_mellin_image_frames(result.data.sai,result.data.usermodu
le,signal_duration,c_2pi) % plot one frame at a time
%plot_sai_mellin_image_frames2(result.data.sai,result.data.usermo
dule,signal_duration,c_2pi) % plot two frames at a time
%plot_sai_mellin_image_frames3(result.data.sai,result.data.usermo
dule,signal_duration,c_2pi) % plot three frames at a time

clear result
```

```
end

end

end % this ends the if plotting
warning on

end % this ends the calculation of the for loop

%%% display computation time

fprintf('\nThe computation time in minutes for each file
was\n\n')
%[a,files] = parameter_file_list; % this function isn't used
anymore
for i = 1:num_files
    [file,dir] = get_parameter_file(i,files);
    fprintf('%s\t%-0.2f\n',char(dir),time(i)/60)
end

fprintf('\nThe total computation time was\n\n')
fprintf('    %-0.2f minutes',sum(time)/60)

fprintf('\n\nThe average computation time was\n\n')
fprintf('    %-0.2f minutes\n\n',(sum(time)/60)/length(time))


end % this ends the if z ~= 4



if z == 4

echo on
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% write Mellin data to HTK format
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
echo off

for i = 1:num_files
tic

global result % so that result can be cleared within the harvest
mellin function

[signal_filename,dir] = get_parameter_file(i,files);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% load some variables that will be needed later
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
all_options.signal.signal_filename = signal_filename;
all_options.signal.start_time=0;
max_delay = all_options.sai.irinosai.maxdelay;
c_2pi = all_options.usermodule.mellin.c_2pi;
% load the wave file
current_directory = pwd;
cd(fullfile(path1,'MSwav_files'));
[y,Fs,nbits,opts] = wavread(signal_filename);
cd(current_directory);
all_options.signal.duration = length(y)/Fs;clear y
signal_duration = all_options.signal.duration;


% load previously computed Mellin data into result
result.data.usermodule =
load_mellin(i,signal_filename,dir,path3);

%%% put PCA here? %%%
%min_frac = 0.05 % minimum fraction variance component to keep
%mellin_image = pca_mellin(min_frac,i,signal_filename,dir,path3)
%%%%%%%%%%%%%%%%%%%%%

% call the harvest_mellin_data function
mellin_image = harvest_mellin_data_htk(result);
%clear result

frame_period =
round(1000*signal_duration/size(mellin_image,2))/1000;


current_directory = pwd;
cd(fullfile(path4,'HTK_files'));
warning off;
mkdir(dir);warning on
[pathstr,name,ext] = fileparts(signal_filename);
filename = fullfile(pathstr,name);
fprintf('%-0.0f writehtk %s\n',i,filename)
writehtk(sprintf('%s.htk',filename),mellin_image',frame_period,9)
cd(current_directory)

clear mellin_image

time(i) = toc;

if rem(i,10) == 0
    pack;
end

end

%%% display computation time
```

```
fprintf('\nThe total computation time was\n\n')
fprintf('    %-0.2f minutes',sum(time)/60)

fprintf('\n\nThe average computation time was\n\n')
fprintf('    %-0.2f seconds\n\n',(sum(time))/num_files)

end % this ends if z == 4
```

Appendix D: Synchronize SAI Matlab Script

```
function [data_final,newSigLen] =
synch_sai(data,strobes,signal_duration,max_delay,sampling_rate)
%
% Written by Lt Jesse Hornback
%

[a NumFrames] = size(data);

% find the strobe points
[Strobes,Strobes_diff] = strobes_est(strobes);
avg_str_diff = mean(Strobes_diff); % find the average time
difference between each strobe
Strobes_diff = [Strobes_diff avg_str_diff]; % account for this
variable being one shorter than Strobes

desired_portion = sampling_rate; % part of the .035 sec window
that we want to take out and keep
if avg_str_diff/desired_portion > 1.375 |
avg_str_diff/desired_portion < 0.75
    fudge_factor = desired_portion/avg_str_diff;
else
    fudge_factor = 1;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% do the uniform sampling here
which_strobes =
sample_strobes(Strobes_diff,signal_duration,sampling_rate); %
find which frames to use at the SR
NumFrames_new = length(which_strobes);
for i = 1:NumFrames_new
    data_sampled{i} = data{which_strobes(i)};
    Strobes_diff_new(i) = Strobes_diff(which_strobes(i));
end
clear data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Get the data and take out the superfluous data
%newSigLen = [];
LenDif = 0;
h = waitbar(0,'Synchronizing SAI Image Data...');
for i = 1:NumFrames_new
    c = struct(data_sampled{i});
    [NumCh SigLen] = size(c.values);
    % find what size the new matrix should be with the
superfluous data removed
    %newSigLen(i) =
round(SigLen*(desired_portion/max_delay)*(Strobes_diff(i)/avg_str
_diff));
```

```
    newSigLen(i) =
round(SigLen*(Strobes_diff_new(i)/max_delay)*fudge_factor); % use
this if desired portion = avg_str_diff
    if size(c.values,2) < newSigLen(i);
        newSigLen(i) = size(c.values,2);
    end
    %cc{i}.values = c.values(:,1:newSigLen(i)); % this is the
original
    cc{i}.values =
c.values(:,1:round((sampling_rate/max_delay)*SigLen)); % this is
to try and get an avg of the length of the sampling rate from
each frame
    waitbar(i/NumFrames,h)
end
close(h)
clear data_sampled

% output to frames
for i = 1:NumFrames_new
    data_final{1,i}=frame(cc{i}.values);
end
```

Bibliography

1. Rabiner, Lawrence and Biing-Hwang Juang. *Fundamentals of Speech Recognition*, pp. 1,339-340, PTR Prentice-Hall, Inc., (Engelwood, NJ), 1993.

2. Weisstein, Eric W. "Mellin Transform." From MathWorld--A Wolfram Web Resource, 2006. http://mathworld.wolfram.com/MellinTransform.html

3. Turner, Richard E., Marc Al-Hames, and Roy Patterson. "Vowel Recognition using the Mellin Image." Centre of the Neural Basis of Hearing, Dept of Physiology, University of Cambridge, 2003.

4. Young, Steve, Gunnar Evermann, Thomas Hain, Dan Kershaw, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, Valtcho Valtchev, and Phil Woodland. "The HTK Book." Cambridge University, (Cambridge, England). 2002.

5. Duda, Richard O., Peter E. Hart, David G. Stork. *Pattern Classification*, pp. 128-138, Wiley-Interscience, (New York, NY), 2001.

6. Kamm, Terri, Hynek Hermansky, and Andreas G. Andreou. "Learning the Mel-scale and Optimal VTN Mapping." Johns Hopkins University, Center for Language and Speech Processing, 1997 workshop (WS97), 1997. http://www.clsp.jhu.edu/ws97/acoustic/reports/KHAMel.pdf

7. Bleeck, Stefan, and Roy Patterson. "*Aim-mat*: An implementation of the auditory image model in MATLAB." 2004. http://www.mrc-cbu.cam.ac.uk/cnbh/aimmanual/Introduction/Introductionframeset.htm

8. Turner, Richard E., Marc A. Al-Hames, David R. R. Smith, Hideki Kawahara, Toshio Irino, and Roy D. Patterson. "Vowel normalization: Time-domain processing of the internal dynamics of speech." in *Dynamics of Speech Production and Perception*, ed. P. Divenyi, IOS Press, (Amsterdam, The Netherlands), (in press.)

9. Anderson, Timothy R. "A comparison of auditory models for speaker independent phoneme recognition", IEEE International Conference on Acoustics Speech and Signal Processing, p. 231, 1993.

10. Patterson, Roy D. "Auditory images: How complex sounds are represented in the auditory system," *Journal of the Acoustical Society of America*, vol. 21, pp. 183-190, 2000.

11. Irino, Toshio and Roy Patterson. "Segregating information about the size and shape of the vocal tract using a time-domain auditory model: The stabilized wavelet-Mellin transform." *Speech Communication*, vol. 36, no 3, pp. 181-203, March 2002.

12. Tung, Y.K. "Uncertainty on Travel Time in Kinematic Wave Channel Routing." Channel Flow and Catchment Runoff Proceedings of the International Conference for Centennial of Manning's Formula and Kuichling's Rational Formula, Wyoming Water Research Center and Statistics Department, University of Wyoming, 1989. http://library.wrds.uwyo.edu/wrp/89-30/89-30.html

13. Guo, Xiaoxin, Zhiwen Xu, Yinan Lu, and Yunjie Pang. "An Application of Fourier-Mellin Transform in Image Registration," The Fifth International Conference on Computer and Information Technology (CIT'05), Shanghai, China, pp. 619-623, 2005.

14. De Sena, Antonio and Davide Rocchesso. "A Fast Mellin Transform with Applications in DAFx," Proc. Of the 7th Int. Conference on Digital Audio Effects (DAFx'04), (Naples, Italy), pp. 65-66, October 5-8, 2004.

15. Cambridge University Engineering Department, Microsoft Corporation. Hidden Markov Model Toolkit. http://htk.eng.cam.ac.uk, December 2002.

16. Garofolo, J.S., L.F. Lamel, W.M. Fisher, J.G. Fiscus, D.S. Pallett, and N.L. Dahlgren. "readme.doc," p1, The DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus (TIMIT), Training and Test Data, NIST Speech Disc CD1-1.1, 1993.

17. Turner, Richard E. "The Perception of Scale in Speech: Vowel Recognition and the Mellin Transform, Part III Project" pp. 4-5, 2003. http://www.mrc-cbu.cam.ac.uk/~tw01/cnbh/teaching/physics_project/documents/PartIII2003.pdf

18. Brookes, Mike. VOICEBOX: Speech Processing Toolbox for MATLAB. Imperial College of Science, Technology, and Medicine, University of London, Oct 2005. http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html

REPORT DOCUMENTATION PAGE						*Form Approved* *OMB No. 074-0188*

1. REPORT DATE *(DD-MM-YYYY)* 23-03-2006	**2. REPORT TYPE** **Master's Thesis**	**3. DATES COVERED** *(From – To)* Sep 2004 – March 2006

4. TITLE AND SUBTITLE Speech Recognition Using the Mellin Transform	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Hornback, Jesse, R., Second Lieutenant, USAF	**5d. PROJECT NUMBER** If funded, enter ENR #
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN), Bldg. 640 2950 Hobson Way WPAFB OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/GE/ENG/06-22
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFRL/HECP Attn: Dr. Timothy R. Anderson 2255 H Street, WPAFB OH 45433-7022 937-255-8817 DSN: 785-8817	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The purpose of this research was to improve performance in speech recognition. Specifically, a new approach was investigating by applying an integral transform known as the Mellin transform (MT) on the output of an auditory model to improve the recognition rate of phonemes through the scale-invariance property of the Mellin transform. Scale-invariance means that as a time-domain signal is subjected to dilations, the distribution of the signal in the MT domain remains unaffected. An auditory model was used to transform speech waveforms into images representing how the brain "sees" a sound. The MT was applied and features were extracted. The features were used in a speech recognizer based on Hidden Markov Models. The results from speech recognition experiments showed an increase in recognition rates for some phonemes compared to traditional methods.

15. SUBJECT TERMS
Hidden Markov Model, Auditory Image Model, Stabilized Auditory Image, Mellin Transform, Hidden Markov Model Toolkit

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Dr. Steven C. Gustafson (ENGE)
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U	UU	75	**19b. TELEPHONE NUMBER** *(Include area code)* (937) 255-3636, ext 4598; e-mail: Steven.Gustafson@afit.edu